Techniques for the Automatic Extraction of Character Networks in German Historic Novels

vorgelegt von

Markus Krug

Würzburg, 2020

Dissertation zur Erlangung des naturwissenschaftlichen Doktorgrades
der Bayerischen Julius-Maximilians-Universität Würzburg

Abstract

Recent advances in Natural Language Preprocessing (NLP) allow for a fully automatic extraction of character networks for an incoming text. These networks serve as a compact and easy to grasp representation of literary fiction. They offer an aggregated view of the text, which can be used during distant reading approaches for the analysis of literary hypotheses. In their core, the networks consist of nodes, which represent literary characters, and edges, which represent relations between characters. For an automatic extraction of such a network, the first step is the detection of the references of all fictional entities that are of importance for a text. References to the fictional entities appear in the form of *names*, *noun phrases* and *pronouns* and prior to this work, no components capable of automatic detection of character references were available. Existing tools are only capable of detecting proper nouns, a subset of all character references. When evaluated on the task of detecting proper nouns in the domain of literary fiction, they still underperform at an F1-score of just about 50%. This thesis uses techniques from the field of semi-supervised learning, such as Distant supervision and Generalized Expectations, and improves the results of an existing tool to about 82%, when evaluated on all three categories in literary fiction, but without the need for annotated data in the target domain. However, since this quality is still not sufficient, the decision to annotate DROC, a corpus comprising 90 fragments of German novels was made. This resulted in a new general purpose annotation environment titled as ATHEN, as well as annotated data that spans about 500.000 tokens in total. Using this data, the combination of supervised algorithms and a tailored rule based algorithm, which in combination are able to exploit both - local consistencies as well as global consistencies - yield an algorithm with an F1-score of about 93%. This component is referred to as the Kallimachos tagger.

A character network can not directly display references however, instead they need to be clustered so that all references that belong to a real world or fictional entity are grouped together. This process widely known as coreference resolution is a hard problem in the focus of research for more than half a century. This work experimented with adaptations of classical feature based machine learning, with a dedicated rule based algorithm and with modern techniques of Deep Learning, but no approach can surpass 55% B-Cubed F1, when evaluated on DROC. Due to this barrier, many researchers do not use a fully-fledged coreference resolution when they extract character networks, but only focus on a more forgiving subset- the names. For novels such as *Alice's Adventures in Wonderland* by Lewis Caroll, this would however only result in a network in which many important characters are missing. In order to integrate important characters into the network that are not named by the author, this work makes use of automatic detection of speaker and addressees for direct speech utterances (all entities involved in a dialog are considered to be of importance). This problem is by itself not an easy task, however the most successful system analysed in this thesis is able to correctly determine the speaker to about 85% of the utterances as well as about 65% of the addressees. This speaker information can not only help to identify the most dominant characters, but also serves as a way to model the relations between entities.

During the span of this work, components have been developed to model relations between characters using speaker attribution, using co-occurrences as well as by the usage of true interactions, for which yet again a dataset was annotated using ATHEN. Furthermore, since relations between characters are usually typed, a component for the extraction of a typed relation was developed. Similar to the experiments for the character reference detection, a combination of a rule based and a Maximum Entropy classifier yielded the best overall results, with the extraction of family relations showing a score of about 80% and the quality of love relations with a score of about 50%. For family relations, a kernel for a Support Vector Machine was developed that even exceeded the scores of the combined approach but is behind on the other labels.

In addition, this work presents new ways to evaluate automatically extracted networks without the need of domain experts, instead it relies on the usage of expert summaries. It also refrains from the uses of social network analysis for the evaluation, but instead presents ranked evaluations using Precision@k and the Spearman Rank correlation coefficient for the evaluation of the nodes and edges of the network. An analysis using these metrics showed, that the central characters of a novel are contained with high probability but the quality drops rather fast if more than five entities are analyzed. The quality of the edges is mainly dominated by the quality of the coreference resolution and the correlation coefficient between gold edges and system edges therefore varies between 30 and 60%.

All developed components are aggregated alongside a large set of other preprocessing modules in the Kallimachos pipeline and can be reused without any restrictions.

Acknowledgements / Danksagung

Zum Entstehen der vorliegenden Arbeit haben viele Personen beigetragen. Dabei gilt mein besonderer Dank an meinen Doktorvater Frank Puppe, der nicht nur stets ein offenes Ohr für Probleme hatte, ich möchte auch insbesondere die angenehmen und spannenden Diskussionen über Forschungsfragen hervorheben, in denen wir immer zu einer sehr guten Lösung gekommen sind, was mir insgesamt eine sehr gute Forschungsumgebung verschafft hat. Des Weiteren möchte ich mich bei Prof. Dr. Fotis Jannidis bedanken, der durch seine bestechende Expertise in der Computerlinguistik und darüber hinaus stets sehr gute Ideen zu unseren Vorhaben beisteuerte.

Für ein angenehmes Arbeitsklima haben auch die zahlreichen Diskussionen und Aktivitäten mit meinen Kollegen des Lehrstuhls für Informatik VI und des Lehrstuhls für Informatik X beigetragen, wofür ich mich auch in diesem Sinne bedanken möchte. An dieser Stelle möchte ich die Unterstützung, die ich durch David Schmidt erhalten habe besonders betonen, ohne seine Unterstützung hätte sich mein Zeitplan sicher noch etwas verzögert. Durch die Zusammenarbeit mit Lukas Weimer und Isabella Reger wurde die Arbeit im Projekt Kallimachos deutlich vereinfacht, so dass die Erstellung der im Rahmen dieser Arbeit entstanden Daten und Tools sehr angenehm verlief.

Darüber hinaus bin ich sehr dankbar für den Rückhalt und die Unterstützung, die ich von meiner Famlie, von meiner Frau und meiner Tochter erhalten habe, so dass auch mal lange Tage im Büro erträglicher wurden.

Würzburg, September 2019

Contents

List of Figures

List of Tables

List of Acronyms

ACE	Automatic Content Extraction
ATHEN	Annotation and Text Highlighting Environment
BERT	Bidirectional Encoder Representations from Transformers
BFGS	Broyden-Fletcher–Goldfarb-Shanno algorithm
BLANC	Bilateral Assessement of Noun-phrase Coreference
CAS	Common Analysis System
CEAF	Constrained Entity-Alignment F-Measure
CoNLL	Conference on Computational Natural Language Learning
CR	Coreference Resolution
CRD	Character Reference Detection
CRF	Conditional Random Field
DROC	Deutsches Romancorpus
DS	Direct speech (utterance)
IAA	Inter-Annotator Agreement
IDE	Integrated Development Environment
IE	Information Extraction
MIRA	Margin-infused relaxed algorithm
MEMM	Maximum Entropy Marcov Model
MUC	Message understanding Conference
NE	Named Entity
NER	Named Entity Recognition
NLP	Natural Language Processing
LDA	Latent Dirichlet Allocation)
LEA	Link-based Entity-aware evaluation metric)
LSTM	Long-short-term-memory
OBIE	Ontology-based Information Extraction

OWL	Web Ontology Language
POS	Part-of-speech
RCP	Rich Client Platform
RHN	Recurrent Highway Network
RUTA	Rule-based text annotation
SMO	Sequential Minimal Optimization
SNA	Social Network Analysis
SVM	Support Vector Machine
UI	User Interface
UIMA	Unstructured Information Management applications
WSJ	Wall Street Journal

1. Motivation

The perennial progress in Artificial Intelligence has reached a state in which most publications do not represent the state of the art by the time they are presented at according conferences. This progress has a severe impact into the daily work of researchers and data scientists in the according field. Instead of creating algorithms by hand (which is often referred to as **G**ood **o**ld **f**ashioned **AI**), which use symbolic representations suitable for custom problems, problems are now tackled using mainly a large amount of data and machine learning. Aside from explicitly telling the program what to do, the engineers design algorithms that learn by experience. This commonality to the way humans learn should however not be overvalued, since according to Judea Pearl, even the most advanced algorithms available at the current time, are "*all stuck there on the level of associations. Curve fitting.*" [Pearl and Mackenzie, 2018]. Natural to the Digital Age, breakthroughs and their associated new technologies spread through the entire world in a very short period of time.

The shift of paradigm took place among a multitude of different tasks and domains. Areas of great success have been in image and signal processing, where first Support Vector Machines [Decoste and Schölkopf, 2002] and models based on Hidden Markov Models [Ephraim and Merhav, 2002] were applied with great success but are nowadays replaced with Deep Learning algorithms [Graves et al., 2013] and [LeCun et al., 1998]. This shift also spread to the Natural Language Processing (NLP) community with the success of neural word embeddings. Interestingly, this shift also had a great influence into how projects with the aim of automating processes are planned, starting with their application to the realization and publication. While a few years ago, it was crucial to have domain experts at hand, you now prioritize data and computation power and instead of designing new, clever algorithms for the problem at hand, existing algorithms are adapted and retrained to perform on the new data.

During this work, it becomes apparent, that even though new technologies are available, they are not the only way to create good algorithms, which is shown for the detection of character references, the automatic detection of speaker and addresses for written direct speech passages and coreference resolution. In all three scenarios, classical rule based approaches managed to perform at least on par with, at the time of writing, most advanced Deep Learning algorithms.

The three most dominant paradigms, that were studied during this thesis are: a) Rule-based algorithms, b) algorithms based on classical, feature based machine learning and c) approaches based on Deep Learning. As author I can present the advantages and disadvantages of the different methods, their requirements and problems but ultimately it is due to the individual data scientist, which methods are to be applied. For rule-based methods, only a small amount of labelled data needs to be available, it can cut the amount of required data that needs to be available during the creation of the algorithm, but especially because the algorithm needs to be evaluated, a data set needs to be available anyway. Rule based approaches require a group of experts which can sense and extract the hidden truth behind the data, formulate it into a symbolic expression (a rule) and implement it into a program. Neither of these tasks are easy, because (and I have gone through this many times myself) language appears to be vastly complex and often times the human intuition[1] is omitting special cases which might end up being forgotten in the realization of the algorithm. Even though it is less considered to be attractive to academics, rule-based approaches have seen progress as well. For one, there are available frameworks which offer rule languages (such as Apache UIMA RUTA [Kluegl et al., 2016] which is part of Apache UIMA or JAPE [Cunningham et al., 1999] which is a part of Gate [Cunningham et al., 2013]). But even if the rule language is not suited or very uncomfortable to solve a problem, there was a huge change in the comfort of general programming. Implementing a rule based algorithm using a modern programming language such as Python, Kotlin, or Rust is much more comfortable compared to an implementation in old fashioned C or C++. Not only simply because the languages have improved, but also very good debugging and profiling mechanism have become intuitive and easy to use, since they usually come integrated in a modern Integrated Development Environment (IDE). A downside of a rule based approach is that the algorithm is relying on either very sophisticated preprocessing or external ressources which might not be available for other domains. Section 3.4 goes over the process, as of how one can decide if a rule based approach is suitable for her goals.

The second paradigm, which is now denoted as classical machine learning, contains traditional algorithms such as the Maximum Entropy (MaxEnt) classifier, the Support Vector Machine (SVM), the Perceptron or any variant of Decision Trees (see section 3.2). What they all have in common and what separates them from traditional rule based approaches is that the engineer needs to only think of the representation of her data and not so much about the algorithm itself. This representation, which is also referred to as *feature extraction* is now at the core of the development process. But the removal of the urge to design a new algorithm for each new problem comes with a downside, the process does now need data to be trained on, data on which sufficient statistics can be extracted in order to fit the parameters of the algorithms. For text mining problems specifically, the Maximum Entropy classifier was successfully applied to a multitude of problems and more and more data sets were made publicly available. Most tasks that appear in text mining come with the advantage, that many features can be identified easily (in the terms of previous or next words or substrings) and can be presented to the algorithm as features, which in turn creates weights for each of the features. For syntactical parsing, this even reached a point, where more than 90% of the runtime of the algorithm was used to extract features - unsurprisingly, since algorithms were pretty much incapable of learning good combinations of features on their own, so that

[1]I am giving a concrete example: In German text, the finite verb of the main clause is usually positioned in the second position. This means, that only a single noun phrase can occur before the finite verb. If this phrase does not contain an article it is very likely to be a name. However in German, abstract nouns that do not refer to names can also be used in this position (such as ”music” or ”art”)

millions of them were added additionally [Kübler et al., 2009]. The natural extension to structured problems of a MaxEnt classifier, called a Conditional Random Field (CRF) [Lafferty et al., 2001], and many derivations of this algorithm dominated the state of the art for about a decade. The inability to identify strong feature combinations on their own yielded improvements in Regularization and feature selection techniques, which could be carried out prior or during the training process [Molina et al., 2002]. Most notably, kernel machines appeared and offered the possibility to comfortably deal with vectors of arbitrary size (by using implicit representations), at the cost of either runtime or memory (or both). Along with their drawback, that their runtime (during application) increases if the set of training data is extended, kernel based classifier tended to yield good performance but they remained to be used for experiments and were not integrated into tools that were delivered to the public (even though a multitude of optimizations exist, e.g. see SVM$_{struct}$ [Tsochantaridis et al., 2004]).

In the 2002 shared task of the CoNNL, Neural Networks (it was not called Deep Learning yet) based on Bidirectional Long Short term Memory layers [Hammerton, 2003] placed last out of 11 contenders, with the Maximum Entropy classifier being in its prime. But times have changed, and the next shift (for text mining) started with the release of *Word Embeddings* in 2011 [Collobert et al., 2011]. While Word Embeddings can be used (with great success, [Jannidis et al., 2017]) by traditional classifiers as well, neural networks are the algorithms which benefitted most of them. Word Embeddings were trained using a vast amount of unlabelled text and create a dense vector (usually ranging from 50 to about 300 dimensions per word) for every word in a corpus. The one which provided the most influence was Word2Vec, but shortly after, additional algorithms, such as GloVe [Pennington et al., 2014], CoVe [McCann et al., 2017] or FastText [Bojanowski et al., 2016] appeared. Having these pretrained resources had a huge impact on the algorithms, instead of pure supervision, which was at the core for almost two decades (obviously there are exceptions, such as the well known Stanford-NER [Finkel et al., 2005]), machine learning has shifted to be semi-supervised, and compared to previous approaches such as Tri-training [Zhou and Li, 2005], Label Propagation [Bengio et al., 2006], Manifold learning [Zhu et al., 2006] or Transductive Learning [Joachims, 1999], this time this approach could be used on a broad array of tasks, simply by starting with these vectors at the beginning of the training process. Having these representations as their input instead of pure 1-Hot vectors, LSTMs started to outperform classical machine learning. On the bright side however, this time it was not a complete replacement of the previous technology, but instead a complement. Recall, that a Maximum Entropy classifier is nothing but a *Softmax* layer with a subsequent *Maximum likelihood* loss, so this time, compared to the classical machine learning only the process of finding suitable features was replaced. However, with the use of Deep Learning, the algorithms were able to recognize good feature combinations, even better, they are even able to detect and make use of nonlinear feature combinations when making decisions (on the downside, this allows them to be great at overfitting as well). In the same manner, CRFs are still widely used in combination with Deep Learning. While LSTMs extract useful information from the input, CRFs make the most of the dependencies in the output labeling. Bidirectional LSTM CRFs [Huang et al., 2015] established themselves as a method for sequence classification. During the last couple of years (and ongoing), the next step of Deep Learning - Multi task Learning (or transfer learning) has become more and more popular. It has always been frustrating to build entire new architectures when the task only changes slightly, this is why the search for a common building block is of great interest. During 2018 and 2019, this building block seemingly was found in the usage of language models. Instead of just starting with one vector per word, the network uses a pretrained language model (which is a neural network itself) to read the data and only add layers on top of the output of these language models (combined with the training process of the new model, this is called fine tuning). In 2018, ElMo [Peters et al., 2018], a language model which is based on multiple stacked BiLSTM layers was released and was selected (by multiple independent sources) to be one of the most influential papers for NLP of that year. The success of these language models quickly established new state of the art results for e.g. automatic translation or question answering. On top, instead of just using LSTM layers, the *Transformer* layer [Vaswani et al., 2017] marks the current state of the art for neural network layers, that deal with textual input and whose task it is to remember and encode the most important bits of a textual sequence into an according dense representation. Using this Transformer layer, BERT [Devlin et al., 2018a], XLNet [Yang et al., 2019], GPT-2 [Radford et al., 2019] and MegatronLM [Nvidia, 2019] are currently seen as the state of the art[2]. The latter, having 8.3 billion (10^9) parameters (this is only about one order of magnitude away of the number of neurons of the human brain).

The question which of these technologies (rule-based approaches, classical machine learning or Deep Learning) are the most fruitful for a different variety of tasks is at the core of this work and the results are manifested in a pipeline suitable for German historic novels. While syntactic preprocessing (such as POS Tagging, Morphology analysis or Parsing) was taken from previously released toolkits, a module for character reference detection, speaker attribution, relation detection and coreference resolution was created. Extending the pipeline by these components required to be active in all areas in the development of the algorithm. These areas comprise the definition of guidelines and the selection of suitable data, which is presented in chapter 4. With guidelines and data at hand, a tool which offers a comfortable way to annotate using these guidelines was realized in ATHEN and is described in chapter 5. Usually a method chapter would follow but since it would vastly exceed 100 pages it is split into five chapters, each dealing with its own task. All chapters come with their own section to denote the related work. Chapter 6 presents the experiments towards a reliable component for the detection of character references, chapter 7 presents experiments for an automatic attribution of a speaker and addressee to direct speech utterances. The adaptation of different coreference systems, as well as their strengths and weaknesses is presented in chapter 8. With both, a character recognition and a coreference module, at hand, nodes for a character network can be represented. Edges of such networks can be modelled using either interactions via communication or interactions that happen during the narrative

[2]However a the time of writing, we were still not able to train such a language model for the German domain of literature, which would enable exciting new experiments

of the text. The latter is presented in chapter 9. This preprocessing is required to extract character networks in a fully automatic manner. New ways of evaluating the character networks are presented in chapter 10. The techniques behind the experiments are presented in chapter 3. This thesis is concluded in chapter 11.

1.1. Context of this Thesis

This thesis, the publications and resources that were created in the process were all part of the Kallimachos project, and as such are made publicly available (if possible). The code that is used to recreate the experiments that are reported in this thesis is all hosted on the Gitlab instance of the department of computer science of the university of Wuerzburg. The resources (ATHEN, automatically created lists, etc..) can be downloaded via a virtual machine that is hosted at the chair of artificial intelligence in Wuerzburg.[3]

[3]http://ki.informatik.uni-wuerzburg.de/nappi/

2. Goals of this Work

This chapter serves as a brief summary of the main contributions of this thesis, as well as a guideline for the *Kallimachos Pipeline*, a standalone tool, which contains the algorithms that were developed during this thesis. Since a large part of this thesis is spent for individual steps of the pipeline in order to automatically extract character networks, this chapter can be seen as an overview of the contents of this thesis.

2.1. Adaptation and Creation of Algorithms for the Domain of German historic novels

The main goal of this thesis is the creation or adaptation of tools required for the automatic extraction of character networks. For this purpose many different experiments have been conducted and the best performing algorithms have been integrated into the *Kallimachos Pipeline* (see section 2.2). In order to create algorithms for different purposes, appropriate data has to be available. Even if the algorithm is working in a purely rule-based fashion, there needs to be data for the quality assessment of the algorithm. Approaches based on machine learning obviously rely on training data (nowadays also called "experience") in order to learn concepts and features that solve a task. During this thesis, the adaptation is performed using either:

- Reannotation and Retraining
- Development of new rule-based approaches
- Adaption of successful algorithms using extra features
- Combination of a trainable component and rule-based approaches.

The above mentioned procedures for the adaptation all produce different results. In order to decide and analyze the strengths and weaknesses of the different algorithms, a component is required which helps to understand the individual results. This component is realized in the form of ATHEN. ATHEN, see chapter 5 was not only the backbone during the creation of the data sets that are used during this thesis (see chapter 4) but also helped to understand and analyze the results of the different algorithms.

2.2. Kallimachos Pipeline

One key result of this work addresses the problem of how the developed algorithms can be executed and shared for reuse. For this purpose the Kallimachos Pipeline, which is a command line interface that is able to configure and launch the algorithms was developed. It is being delivered in an online repository [1] and consists of two files. The first one is the executable, which is a *.jar*-file containing all dependencies and models that are required for a seamless execution of a preconfigured NLP pipeline. The other file is an initiator file, called ”preprocessing.ini”. The user can configure her pipeline in there by simply commenting and uncommenting options. All results follow the data structures that are contained in the typesystem which is presented in appendix C. At the current state, the following engines are included and can be used:

Tokenization A wrapper of the Apache OpenNLP [Baldridge, 2005] tokenizer, which is basically a Maximum Entropy classifier that slides through the text and predicts a label for each character. The resulting labels are converted into tokens in a subsequent step.

A rule-based tokenizer, developed throughout the course of this thesis, developed specifically to be able to yield reasonable quality on literary texts, for more details refer to section 4.9.1.

---

inputs: Raw text.
outputs: Annotations of type **POSTag**.

Sentence Splitting Similarly there is a component of the Apache OpenNLP for sentence splitting of German texts. Furthermore a rule-based component which performs sentence splitting and builds upon the results of the previous tokenizer is contained in the pipeline. It is therefore expected to work best, when combined with the rule-based tokenizer. A detailed description of the algorithm is provided in section 4.9.1.

---

inputs: Raw Text, Tokens.
outputs: Annotations of type **Sentence**.

POS Tagging Currently, three different POS Tagging engines are provided. The first one is the component that comes with OpenNLP. The second POS-Tagger is delivered with the Mate-Toolkit [2]. The third option, that is recommended for use is the TreeTagger [Schmid, 1995]. All three taggers built upon previous tokens and sentences and enrich the tokens with the **POSTag** feature. The taggers produce their output using the STTS-Tagset (see A.1).

---

inputs: Tokens (as POSTag annotations), Sentences.

[1] `http://ki.informatik.uni-wuerzburg.de/nappi/NLP-preprocessing`
[2] `https://code.google.com/archive/p/mate-tools/`, accessed 20.05.2020

outputs: Fills the feature **POSTag** in the POSType.

Paragraph Splitting Splitting the text into paragraphs is an important task for subsequent analysis, e.g. as by the speaker attribution module. There are two modules, the first one, *Simple_Kernkorpus_Paragraph_Detection*, uses heuristics (such as the amount of whitespace in between sentences) to create paragraphs. The second algorithm is just creating paragraphs by splitting the text at line breaks. This is sufficient for the data that originated TextGrid but might not be for different domains or origins.

---

inputs: Tokens, Sentences, raw text.
outputs: Annotations of type **Paragraph**.

Morphology There are currently three modules integrated that can produce morphological information. All three components are trained on the TIGER-corpus released by [Brants et al., 2002]. The use of the RFTagger [Schmid and Laws, 2008] is recommended for this stage since it produces reliable output, is fast and probably even more reliable (in terms of its POS-Tags) than the TreeTagger. It is also noteworthy, that inside the model, after decompression, there is a huge German lexicon which contains millions of different conjugations of different words. Since the models are trained on the TIGER corpus, they produce information about the syntactical gender, the number, the person, the case and the POS-Tag.

---

inputs: Tokens, Sentences
outputs: Annotations of type **RFTagType**, filled with the feature **Tag** which contains one label of the Tagset produced by the RFTagger.

Lemmatizer At the time of writing, two different algorithms that produce the lemma of a word are available. The first originates from the Mate-Toolkit and the second algorithm is part of the TreeTagger. Throughout this thesis, the component of the TreeTagger was utilized and produced realiable results.

---

inputs: Tokens, raw text
outputs: It fills the feature **Lemma** of the type **POSTag**.

Chunker Chunking (also called *shallow parsing*) is currently only supported via the TreeTagger. The reason behind this is that the corpus on which this model was trained is kept private (this was stated in an email conversation with the author of the TreeTagger). This model produces chunks in a form, so that no chunk contains more than one noun (which means that genitive phrases are always split into more than one chunk). The algorithm produces three types of chunks, noun phrases, verb phrases and prepositional phrases.

---

inputs: Tokens, POS-Tags
outputs: It creates annotations of the type **Chunk** and stores the type of the chunk in the feature **ChunkType**.

Parser The pipeline does currently support five different parser. Three of those are constituency parser, and the other two are dependency parser. A constituency parser generates phrases that are connected via the features **parent** as well as **children** in order to navigate the syntax tree. For later components in the pipeline, currently only the coreference resolution makes use of the generated phrases. It uses these phrases in order to sort candidates for the pronominal resolution.

---

inputs: Tokens, POS-Tags
outputs: It creates annotations (which represent phrases) of the type **StanfordParse** and stores the children and parent phrases in the according features.

Both, the parser provided by the Stanford university as well as the parser provided by the university of Berkeley are trained on newspaper text. The third constituency parser was created to work for the domain *körperliche Untersuchung* for German medical discharge letters during an internship which I supervised.

The two other parser are dependency parser and create new annotations on the token level. Each of these annotation has access to the head (which is an annotation as well) and the dependency relation in the NEGRA scheme (as seen in appendix A.2). The parser of the Mate Toolkit [Bohnet, 2010] is a trained graph based parser and the ParZu parser [Sennrich et al., 2009] is a rule-based parser, that makes use of the CKY-algorithm for inference [Kasami, 1966]. It is noteworthy, that in order to be able to use the ParZu parser, Docker [Anderson, 2015] needs to be installed on the computer. The dependency parser creates the following annotations:

---

inputs: Tokens, POS-Tags
outputs: It creates annotations (which represent dependency links) of the type **DependencyParse** and stores the head (which is another annotation or null, if the token is the root of the sentence) as well as the dependency relation in the according features.

Speaker Attribution Even though a multitude of different algorithms have been created and evaluated during the course of this thesis (including one for the detection of quotations, even when no marker are available [Tu et al., 2019]), only one algorithm was integrated and is available for the detection of speaker and addressee for direct speech utterances. The algorithm detects the utterances on its own by the usage of simple regular expressions. It continues to classify each expression into either a direct speech or *other*. For the utterances that were determined to be direct speech, a speaker and an addressee is assigned (or null if none could be determined). The algorithm makes extensive use of the coreference module in order to produce consistent results (e.g. to prevent utterances where speaker and addressee are the same entity). The speaker and addressees are selected from previously detected character references and as such are stored as (references to) annotations.

---

inputs: Tokens, POS-Tags, Morphology, Character References, Dependency Parse Information
outputs: It creates annotations of the type DirectSpeech

The algorithm is described in more detail in chapter 7.

Character Reference Detection For the detection of the spans that refer to literary characters, two modules are available. The first module, referred to as *RulebasedNE*, is a pure rule-based algorithm. The second module, named *KallimachosTagger* is a combination of the rule-based system with a Maximum Entropy classifier to produce optimal outputs. Both modules expect the outputs of all previous modules except for the Speaker Attribution. They produce spans of character references, and assigns the information of a coarse type for the character, whether the reference is a name (labelled as "core"), a noun phrase (labelled as "appellativ") or a pronoun. More details regarding these algorithms is provided in chapter 6.

---

inputs: Tokens, POS-Tags, Morphology, Dependency Parse Information, Chunks
outputs: It creates annotations of the type NamedEntity and sets the feature NEType

---

Coreference Resolution The core of the pipeline is the coreference resolution module. This algorithm uses the previously annotated information as best as possible and assigns identifier to the character references which serve as input. The same identifier depicts the membership to a common entity. First, the algorithm attaches more fine-grained information about the character references(see 8.2.2) and then applies a set of rules (called sieves) one after another onto the text. A more detailed description can be found in chapter 8

---

inputs: Tokens, POS-Tags, Morphology, Dependency Parse Information, Chunks, Character References, Attributed Speaker
outputs: Sets the feature *ID*, as well as the features *Construction* and *Gender* of an annotation of type NamedEntity

---

Relation Detection At the current point, there are two modules for the detection of relations between characters included, both algorithms are described in more detail in chapter 9. The first module, called *MaxEntRelationDetection* is a combination of a rule-based algorithm and a Maximum Entropy classifier, similar to the procedure that was applied for the Kallimachos Tagger. This module produces four types of relations, *family relations*, *social relations*, *professional relations* and *love relations*. The second module, named *HierMaxEntInteractionDetection* was created during a masters thesis and serves the purpose to detect interactions between character references that can subsequently be used for the creation of character networks.

---

inputs: Tokens, POS-Tags, Morphology, Dependency Parse Information, Chunks, Character References and Attributed Speaker
outputs: Creates Annotations of the type *Relation* and sets the feature *Type*, to further describe the relation.

Ontology Based Information Extraction There is currently one module that allows the extraction of attributes using a user defined ontology and previously annotated dependency information. A more in depth explanation of this component can be found in section 8.2.2.

---

inputs: Tokens, Dependency Parse Information, Chunks
outputs: Creates Annotations of the type *IEEntity* as well as annotations of the type *Relation* to indicate a binary relationship between two entities.

2.2.1. Experiments concerning Runtimes and Requirements of the different Algorithms

Since the runtime of the pipeline is in the magnitude of minutes, this section shows the time of different components, when applied on different novels. This should help, when configuring a pipeline on a personal computer. The distribution of runtimes for the typical execution of the Kallimachos Pipeline is shown in figure 2.1[3]. This distribution was obtained after applying the pipeline to the following five novels:

1. Louise Aston - *Lydia*
2. Theodor Fontane - *Effi Briest*
3. Theodor Fontane - *Mathilde Möhring*
4. Johann Wolfgang von Goethe - *Die Wahlverwandschaften*
5. Karl May - *Und Friede auf Erden!*

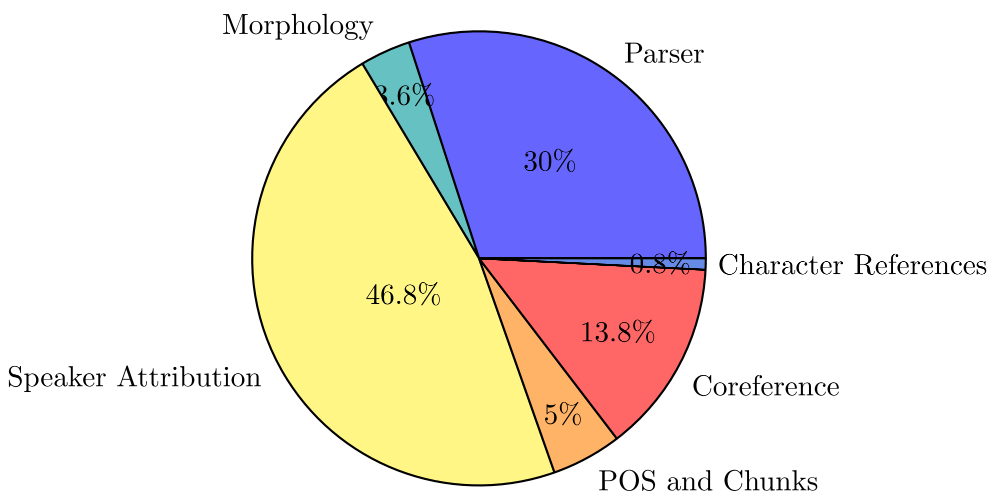


Figure 2.1.: Typical runtime distribution, when applying the Kallimachos Pipeline until Coreference Resolution. Tokenization, sentence splitting and the detection of paragraphs are all negligible when it comes to the runtime. The total runtime for five novels was about an hour when executed on an Intel Core i7-4770

In order to provide a more detailed analysis of the required resources, all components were executed on the five novels, and their runtimes plotted. This overview can be seen in figure 2.2

[3]This distribution was obtained using the version of the Kallimachos Pipeline as of 19.08.2019

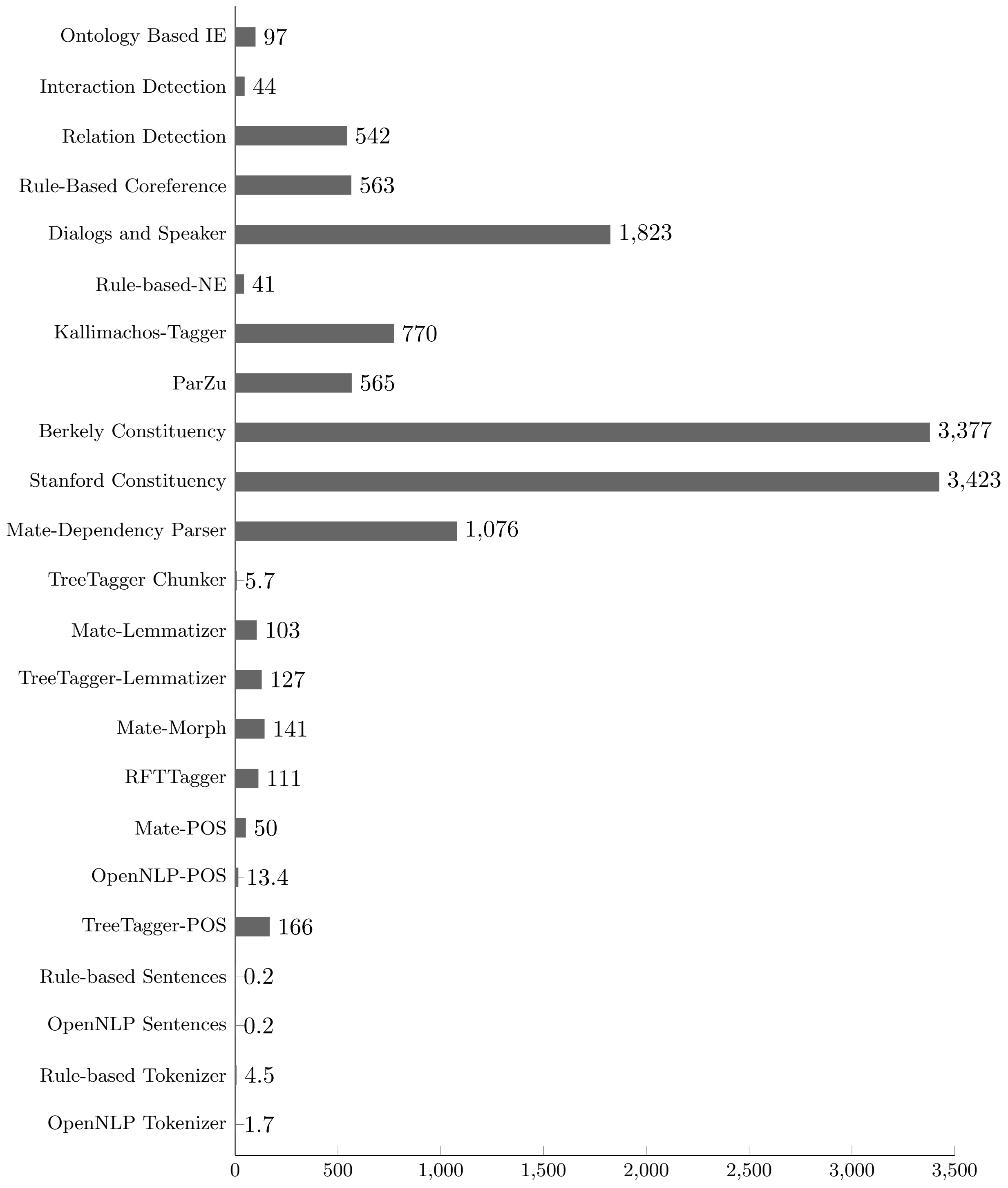


Figure 2.2.: The individual sums of the runtimes of the components that are integrated in the Kallimachos pipeline (in seconds), when applied on the five aforementioned documents.

2.3. Contributions

In order to create the aforementioned modules, this thesis has to provide contributions on all ends of the development of automatic algorithms. This involves the definition of guidelines for the annotation of character references, speakers and addressees, as well as relations and interactions. I have to mention, that I did not define these guidelines by myself, but they were instead created in a group effort, where I want highlight the participation of Prof Dr. Frank Puppe, Prof Dr. Fotis Jannidis and Lukas Weimer. After guidelines were created and corresponding data for the annotation was selected, it requires the use of a suitable tool to conduct the annotation with. This resulted in the creation of the annotation environment for relations which is based on Active Learning as well as the creation of ATHEN. ATHEN has grown to be a huge software comprising more than 100.000 lines of code, and was ultimately used to annotate about 500.000 tokens for CRD and coreference, as well as about 100.000 tokens with relations and interactions. ATHEN was extended by the project "Redewiedergabe" and was ported into a Web version which is currently still in development. This data serves as an excellent basis for the development and adaptation of algorithms for automatic components. This resulted in the Kallimachos-Tagger, which is a combination of a rule-based and a machine learning system, as well as a Coreference Algorithm, based on the rule-based system of Stanford which is adapted to German novels and exploits, matches by the usage of meta data as well as the integration of a speaker attribution. For the automatic extraction of relations between characters, a dedicated rule-based approach as well as a novel kernel for the Support Vector Machine was proposed. Yet again, the combination of the rule-based approach with a classical machine learning approach provided the best results on most labels. A modern justification of rule-based approaches (as opposed to approaches based on machine learning or Deep Learning) is presented and gives insight why rule-based approaches can still capture aspects of the data that can not be modelled by machine learning. For most applied techniques, an error analysis was carried out, so that the individual problems could be analyzed and new methods to tackle problems (especially the coreference resolution) can be developed (I present multiple ways for the improvement in my outlook in section 11.2).

This work provides results on diverse frontiers, rule-based approaches, approaches based on classical machine learning and result based on Deep Learning. Aside from new data sets, adapted algorithms, a new annotation environment, this work also proposes new ways to evaluate coreference algorithms and social networks. For the evaluation of the coreference metrics a new metric was proposed, which offers more insight into the actual problems and does not simply provide a quantitative score.

The evaluation of automatically extracted character networks was enriched by methods that involve the usage of summaries to the novels and promote the usage of evaluation scenarios based on rankings compared to evaluation scores that yet again are only based on numbers. Since these numbers are only converted into the size of a node or an edge in the network, these are more or less mainly cosmetic aspects and ranking pose more weight onto the correctness of their relative numbers instead of absolute values, since these can be seen in the network.

3. Preliminaries

This chapter introduces most of the concepts that are used throughout this thesis. Since this thesis proposes a new metric for coreferences, that improves upon shortcomings of previous established metrics, the existing metrics are presented in great detail so that the reader should be able to follow the argumentation (3.1). As this work compares rule-based, classical machine learning and deep learning on a multitude of different tasks, it starts with an introduction of a taxonomy, in which the different approaches can be seen. The most prominent classical machine learning algorithms were introduced, starting with the Perceptron in section 3.2.1 and extends the Perceptron optimization problem to end up at Support Vector Machines. This shows that these algorithms are very common in their principle of optimizing a linear function, but they utilize different constraints. In this setting, I am providing a detailed derivation of the MIRA update as it is used throughout this thesis in order to distinguish it from different update mechanisms for the Perceptron. Section 3.2.4 briefly introduces Deep Learning as a complex function that is represented using a computation graph, which can be utilized for the determination of the gradient using an algorithm similar to the backpropagation algorithm. This brief introduction is mainly done in order to lay the details to understand a neural conditional random field, as it is used throughout this work. At the time of writing, neither Tensorflow[1] nor PyTorch[2] come with an implementation of neural CRFs beyond a linear chain, which wastes a lot of modelling power of conditional random fields in general. This work introduces a CRF as a collection of templates, each template having their own learnable parameters, that can be decoded efficiently using the forward-algorithm. In section 3.3 this work introduces the general approach that was taken throughout this work whenever a rule-based algorithm was developed. The chapter is concluded in section 3.4 with a high-level, subjective comparison of rule algorithms against machine learning algorithms. It illustrates why a rule-based algorithm can still perform on par with machine learning.

3.1. Performance Metrics

This section goes briefly over the performance metrics used in this thesis. The performance metrics for coreference resolution are accompanied by an example to illustrate their characteristics, since they are more complicated.

[1]`https://www.tensorflow.org/`, accessed 20.05.2020

[2]`https://pytorch.org/`, accessed 20.05.2020

3.1.1. Performance Metrics for Labelwise Evaluation

If the goal is to evaluate an unstructured classification problem, such as one that arises during a regular multi-class classification problem (e.g. for the speaker attribution, where only two classes are available), the most suitable performance metrics that were applied are the accuracy and the **labelled F1-score**. The accuracy in this manner is just counting the amount of correct predictions into a variable *correct* and setting it in relation to the total amount of predictions to be done. In the case of a simple binary prediction, where usually just one class is of interest and the second class is considered to be the background label, the amount of correct predictions can be further divided into two categories, namely the *true positives* and the *true negatives.* Each confusion then accounts for a *false positive* if a label is falsely predicted, as well as for a *false negative* which means that the correct label is missing in the system solution.

The second and arguably the more useful way for evaluation is by the use of the labelled F1-score. In this setting, three individual counters (tp, fp, fn) are introduced for every available label and the aggregated results are then reduced into Precision, Recall and F1-score. This metric allows an interpretation of the results of an algorithm by using a confusion matrix. This matrix contains the counts of the pairwise confusions between the labels and therefore helps the engineer to improve and direct her algorithms even without an in depth error analysis. This attribute of a performance metric is very important. I am mentioning this here because none of the current standard evaluation metrics for coreference resolution offer this (see section 3.1.3) There is also a way to reduce the individual (per label) F-scores into a single evaluation score. This can be done by averaging the F1-score of the individual classes. The average might respect the distribution of the labels in order to not be dominated by small classes. The calculation of Precision, Recall and F1-score is as follows:

$$\begin{aligned}\text{Precision} &= \frac{tp}{tp + fp}\\ \text{Recall} &= \frac{tp}{tp + fn}\\ \text{F1} &= \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}\end{aligned}$$

I am going to mention the distinction between the *micro-F1* and the *macro-F1.* If an evaluation is to be performed over some sort of context (such as a document), then there are at least two possibilities to aggregate the counts of the individual documents. The first way is to just sum the counters between documents, to get final aggregated counts that can be used to calculate the F1-score. This setting is referred to as the micro F1-score. The second possibility is to count in the individual documents, and calculate a Precision, Recall and F1-score for each individual document. These scores are subsequently averaged to create the macro-F1. It is important to mention the nature of the metric because they offer different ways to be interpreted. The micro score ignores the documents and therefore provides a statement about the total quality on the test data. The macro score on the other hand treats big documents equally as important as small documents and might therefore show large differences when compared side by side with the micro score.

3.1.2. Performance Metric for the Evaluation of Spans

Character reference detection (see chapter 6), among many other tasks in the NLP, is predicting spans of tokens and if a label is required, then the label has to be predicted for this span as well. The general way to treat this problem is to predict (in the same manner as for the unstructured case) a label of the set $\{B, I, O\}$. The first entry (B) refers to the begin of a span, the second (I) refers to an intermediate token of a section and the last entry (O) means that the token is outside any span of interest. Predicting these labels for individual tokens allows for a postprocessing algorithm to calculate the spans (in the means of token indices). In this manner, a list of spans can be calculated for the system solution (produced by an algorithm) as well as for the gold data. Having this data structure at hand, the calculation of the **Entity-F1** proceeds as follows: First, the amount of true positives is determined. A true positive occurs if there is a span in the gold spans that is **equal** to a span in the system spans. Equality for the **Unlabelled** setting is defined by comparing the token offsets (usually referred to as begin and end), while for the **Labelled** setting on top of the spans, also the label of that span is taken into account. This strict definition means, that the system gets no reward if it only predicts a span partially, instead is punished by both - a false positive and a false negative. The distinction between a micro and a macro score holds in the same way as for the unstructured case.

If the spans appear to contain many tokens, then it might be more beneficial to use a different labeling schema (for the classifier) for the tokens instead of the B-I-O encoding. There has been work for the comparison the individual encoding schemes (e.g. see [Sassano and Utsuro, 2000]). I would recommend to encode short spans (up to a length of about 5 tokens) using the B-I-O scheme. If it can be guaranteed, that no two spans are direct neighbours (in terms of tokens), then I would suggest to switch to an I-O encoding. If the spans exceed five tokens in many instances, then I would always consider switching from a setting that predicts the spans, to a setting that predicts the boundaries.

3.1.3. Performance Metric for the Evaluation of Coreference Clusters

Evaluating the score of an automatic coreference resolution is not straight forward. In essence, CR performs a clustering over mentions (in this work the mentions are character references) and the evaluation procedure can not rely on all mentions to be available in both, the solution produced by the system and the gold clustering. In general there are two different views on the comparison between system and gold clusterings. The first view reduces a cluster of a clustering into a set of links (a link is an edge between two mentions) and calculates evaluation scores based on the amount of correct links that are contained in the system clustering (e.g MUC, see section 3.1.3), while the second view calculates the evaluation score based on the overlap of individual clusters of the gold and system clustering (e.g. B-Cubed, see section 3.1.3). The reduction of a cluster into a set of links can either be done minimalist, which means that for a cluster with N mentions, only $N-1$ links are created, or the reduction is done by creating all possible edges between mentions in a cluster. The two different ways of how a cluster is reduced also offer room for interpretation: In the minimalist scenario, big clusters (such as a cluster of a main character that appears many times throughout a novel) are reduced to a linear amount of links, while in the full scenario a quadratic amount of links is created. If both - small clusters and big clusters are available, then the minimalist approach does not value a correct resolution of a big cluster higher than an equal amount of small clusters. Since the view based on edges has (among others) the aforementioned issues, more metrics that are based on a different view on the clustering were created. The second class of evaluation metrics are based around measuring the pairwise overlap of clusters. This view allows a more fair evaluation of documents that have clusters of many different sizes.

Since the explanation of the performance metrics involves calculations between a gold and a system clustering, the following notations are introduced for this section:

- G_i represents a gold entity
- $\mathcal{G}$ represents the set of all gold entities
- S_j represents a system entity
- $\mathcal{S}$ represents the set of all system entities

MUC

The MUC performance metric was published in 1995 by Villain et al. [Vilain et al., 1995] and evaluates the compatibility of a clustering by reducing the cluster into links using the minimalist approach. That said, in order to connect all mentions m for an entity G_i, an minimal amount of $|G_i|-1$ of links is required. The MUC metric then continues to count all links that are faulty, when the links of the gold clustering and the links of system clustering are compared. The determination of all faulty links is done in a rather elegant way using *partitions* of an entity. A partition of a gold entity G_i is the set of clusters of the system clustering that contain at least one mention of the gold cluster G_i:

$$p(G_i, \mathcal{S}) = \{G_i \cap S_j | j = 1, \ldots, |\mathcal{S}|\} \cup \bigcup_{m \in (G_i \setminus \mathcal{S})} \{\{m\}\}$$

If for example a gold entity G_i consists of the four mentions $\{A, B, C, D\}$, one system entity contains the mentions $\{A, B\}$ and the other two mentions C and D are located in two separate system entities, then the partitioning of the gold entity G_i among the system clustering S is calculated as: $p(G_i, \mathcal{S}) = \{\{A, B\}, \{C\}, \{D\}\}$ The partitioning of the gold entity is then used to calculate the amount of faulty links between the gold and the system clustering. If the partitioning contains k system entities, then $k-1$ additional links would be necessary in order to connect them into a single entity (for the example two additional links would be necessary). These faulty links are also considered to be *missing links*.

These statistics can then be used in order to calculate the Recall and Precision of the MUC metric. The Recall is calculated as follows:

$$R = \frac{\sum_{i=1}^{|\mathcal{G}|}(|G_i| - |p(G_i, \mathcal{S})|)}{\sum_{i=1}^{|\mathcal{G}|}(|G_i| - 1)}$$

This formula means, that for all gold entities, the amount of correct links are set in relation to the minimal amount of links required to connect the mentions in a cluster.

The Precision is obtained by exchanging the system and the gold clustering. Missing links can then be interpreted as those links that have been predicted on top of the required and correct links. Therefore the Precision is defined as:

$$P = \frac{\sum_{i=1}^{|\mathcal{S}|}(|S_i| - |p(S_i, \mathcal{G})|)}{\sum_{i=1}^{|\mathcal{S}|}(|S_i| - 1)}$$

The F1-score is, as usual, the harmonic mean between Precision and Recall.

$$F_1 = \frac{2PR}{P+R}$$

Example: Given the gold entities $\{1,2,3,4,5\}, \{6,7\}, \{8,9,A,B,C\}$ as well as the entities produced by the system $\{1,2,3,4,5\}, \{6,7,8,9,A,B,C\}$. The partitioning of the gold entity $\{1,2,3,4,5\}$ is then given by $\{\{1,2,3,4,5\}\}$, the partitioning of $\{6,7\}$ is $\{\{6,7\}\}$ and the partitioning of the entity $\{8,9,A,B,C\}$ is $\{\{8,9,A,B,C\}\}$.

This allows for a calculation of the Recall:

$$R = \frac{(5-1)+(2-1)+(5-1)}{(5-1)+(2-1)+(5-1)} = 1$$

The partitioning of the system entity $\{1,2,3,4,5\}$ is $\{\{1,2,3,4,5\}\}$, and the partitioning of $\{6,7,8,9,A,B,C\}$ is $\{\{6,7\},\{8,9,A,B,C\}\}$. This allows the calculation of the Precision:

$$P = \frac{(5-1)+(7-2)}{(5-1)+(7-1)} = \frac{9}{10}$$

and the calculation of the F$_1$-score:

$$F_1 = \frac{2 \times 1 \times \frac{9}{10}}{1 + \frac{9}{10}} = \frac{18}{19}$$

Problems of this metric: One problem of the MUC metric is, as previously mentioned, that it is penalizing every faulty link equally. If an algorithm would connect the gold entities $\{1,2,3,4,5\}$ and $\{8,9,A,B,C\}$ instead of the gold entities $\{6,7\}$ and $\{8,9,A,B,C\}$, the resulting system entities would be $\{1,2,3,4,5,8,9,A,B,C\}$, $\{6,7\}$. The Recall would remain the same, but for the precision one would get:

$$P = \frac{(10-2)+(2-1)}{(10-1)+(2-1)} = \frac{9}{10}$$

.

This means, that all three performance metrics show the same values and therefore do not differ between missing a link in a more important entity or in a smaller one. Arguably the second mistake should be weighted more severe as the first one.

A different issues arises, when considering entities that consist of a single mention (so called singletons). Since during the reduction to links, no single link is created, singletons do not influence the result at all. For the Recall and the F1-score, its contribution would be 0, while the Precision is even incalculable because it would require a division by zero.

B-Cubed

Bagga and Baldwin [Bagga and Baldwin, 1998] presented the B-Cubed performance metric in 1998 with the purpose to solve some of the aforementioned issues of the MUC metric. The B-Cubed metric is the first metric which calculates the quality based on the overlap of gold and system cluster and does not reduce the clusters to links as the MUC metric. This procedure does immediately eliminate the problem of an according influence of singleton clusters in the MUC metric.

The Recall is now defined by considering every gold **mention** inside the entities. The first definition is to calculate the Recall of a mention m:

$$R(m) = \frac{|G_i \cap S_j|}{|G_i|}$$

This requires the according gold and system entities of the mention m and calculates their overlap (based on common mentions), divided by the amount of mentions in the gold entity. If no system cluster is available, then the Recall is 0 as well. In order to obtain the Recall of an overlapping pair of gold and system entities, the Recall for each mention m, that is contained in both entities is summed:

$$R(G_i, S_j) = \sum_{m \in (G_i \cap S_j)} R(m) = \frac{|G_i \cap S_j|^2}{|G_i|}$$

Finally the Recall for the entire clustering is calculated by summing the Recall of all (possible) pairs of system and gold entities and by dividing by the amount of gold mentions $\mathcal{M}_g$:

$$R = \frac{\sum_{i,j} R(G_i, S_j)}{|\mathcal{M}_g|} = \sum_{i,j} \frac{|G_i \cap S_j|^2}{|G_i|\,|\mathcal{M}_g|}$$

Similarly as in the MUC score, the Precision is obtained by exchanging the position of gold clustering and system clustering:

$$P(m) = \frac{|G_i \cap S_j|}{|S_i|}$$

$$P(G_i, S_j) = \sum_{m \in (G_i \cap S_j)} P(m) = \frac{|G_i \cap S_j|^2}{|S_j|}$$

$$P = \frac{\sum_{i,j} P(G_i, S_j)}{|\mathcal{M}_s|} = \sum_{i,j} \frac{|G_i \cap S_j|^2}{|S_j|\,|\mathcal{M}_s|}$$

And the F1-score is yet again the harmonic mean:

$$F = \frac{2PR}{P+R}$$

Example: Reconsidering the same example as for the MUC score, given the gold entities $\{1,2,3,4,5\}, \{6,7\}, \{8,9,A,B,C\}$ as well as the system entities $\{1,2,3,4,5\}$ and $\{6,7,8,9,A,B,C\}$. The Recall sums to 1, because the intersection between gold entities and system entities does always contain all the mentions of the gold entities.

$$R = \frac{\frac{5^2}{5} + \frac{2^2}{2} + \frac{5^2}{5}}{12} = 1$$

When calculating the Precision, for the system entity: $\{6,7,8,9,A,B,C\}$, there are two intersections with different gold entities, more precisely between $\{6,7\}$ and $\{8,9,A,B,C\}$ (the system entity $\{1,2,3,4,5\}$ has yet again just a single overlap with the according gold entity).

$$P = \frac{\frac{5^2}{5} + \frac{2^2}{7} + \frac{5^2}{7}}{12} = \frac{5 + \frac{29}{7}}{12} = \frac{16}{21}$$

Taking these values results in an F1-score of:

$$F = \frac{2 \times 1 \times \frac{16}{21}}{1 + \frac{16}{21}} = \frac{32}{37}$$

B-Cubed gets rid of the main problems of the MUC metric but introduces its own weak points. The first one is that its Recall is not bounded by 1. If for example a system mention is part of more than one entity, this might occur. (In theory this is also possible for the Precision but this would require that a gold mention is part of more than one entity). The following example shows how a such a Recall can be attained: Having a single gold entity $\{a,b,c\}$ as well as three system entities $\{a,b\}, \{a,c\}, \{b,c\}$.

As a is part of two intersections between gold and system entities, the Recall for the mention a is a sum of two terms, in both cases, the term is $\frac{2}{3}$, yielding a total Recall for the mention a of $\frac{4}{3}$. The same holds for b and c. The total Recall is then

$$R = \frac{\frac{4}{3} + \frac{4}{3} + \frac{4}{3}}{3} = \frac{4}{3} > 1$$

The next issue with the B-Cubed metric is that its results are very sensible towards simple baselines. Creating a system solution where all mentions are located in a single entity results in a Recall of 1. The other extreme, having a separate entity for every mention results in a Precision of 1

CEAF

Due to the weak points of previous available metrics, Luo developed the CEAF (Constrained Entity-Alignment F-Measure) metric in 2005 [Luo, 2005]. For its calculations, it builds upon the ideas of the B-Cubed metric, that the evaluation score should be determined by cluster overlaps. But in contrast to B-Cubed, CEAF does not consider all pairings between system entities and gold entities but instead only considers one-to-one mappings between the clusters of the two clusterings.

In the beginning, the minimum amount of entities between system and gold clustering are calculated:

$$m = \min\{|\mathcal{G}|, |\mathcal{S}|\}$$

Then the set of all one-to-one mappings $\mathcal{A}_m$ can be expressed as: Defining $\mathcal{G}_m \subset \mathcal{G}$ and $\mathcal{S}_m \subset \mathcal{S}$ to be sets, consisting of m gold or system entities $|\mathcal{G}_m| = m$ and $|\mathcal{S}_m| = m$. Let $\mathcal{A}(\mathcal{G}_m, \mathcal{S}_m)$ be the set of all one-to-one mappings between $\mathcal{G}_m$ and $\mathcal{S}_m$, more formally:

$$\mathcal{A}(\mathcal{G}_m, \mathcal{S}_m) = \{a : \mathcal{G}_m \to \mathcal{S}_m\}$$

$$\mathcal{A}_m = \bigcup_{\mathcal{G}_m, \mathcal{S}_m} \mathcal{A}(\mathcal{G}_m, \mathcal{S}_m)$$

The next ingredient required for the CEAF performance metric is a similarity function between two entities: $\phi(G, S)$. The similarity function may not return negative values, which means that $\phi(G, S) = 0$, only if G and S do not show any similarities.

Two examples for such a similarity function were presented in [Luo, 2005]. The first one (ϕ_3) calculates the overlap based on the cardinality of the intersection of the gold and the system entity, the second metric (ϕ_4) uses the relative amount of common mentions between G and S.

$$\phi_3(G, S) = |G \cap S|$$

$$\phi_4(G, S) = \frac{2\,|G \cap S|}{|G| + |S|}$$

Applying this similarity function to all pairs of system and gold entities results in a matrix of similarity scores. From this matrix, the mapping a needs to be selected, so that the sum of all similarities of the individual pairs is maximized.

$$\Phi(a) = \sum_{G \in \mathcal{G}_m} \phi(G, a(G))$$

One possibility to solve this optimization problem is by applying the Hungarian algorithm (also called Kuhn-Munkres algorithm) [Kuhn, 1955] in order to determine the best mapping a^*. Afterwards, the Recall, Precision and F1-score are defined as follows:

$$R = \frac{\Phi(a^*)}{\sum_i \phi(G_i, G_i)}$$

$$P = \frac{\Phi(a^*)}{\sum_j \phi(S_j, S_j)}$$

$$F = \frac{2PR}{P + F}$$

The numerator contains the amount of common mentions (according to the mapping function ϕ_3) and the denominator represents the amount of the gold and system mentions respectively. This is why the CEAF score using ϕ_3 is also called **mention-based** and is denoted as CEAF$_m$. If the mapping function ϕ_4 is used, the the resulting metric is also denoted as **entity-based**, or short CEAF$_e$

Example: Using the running example: a^* maps $\{1, 2, 3, 4, 5\}$ to $\{1, 2, 3, 4, 5\}$ and maps $\{8, 9, A, B, C\}$ to $\{6, 7, 8, 9, A, B, C\}$. $\{6, 7\}$ is not considered as there are only two system entities. For $CEAF_m$ this results in:

$$R = \frac{5 + 5}{5 + 2 + 5} = \frac{10}{12}$$

$$P = \frac{5 + 5}{5 + 7} = \frac{10}{12}$$

$$F = \frac{2 \times \frac{10}{12} \times \frac{10}{12}}{\frac{10}{12} + \frac{10}{12}} = \frac{5}{6}$$

And accordingly for $CEAF_e$:

$$R = \frac{1 + \frac{10}{12}}{1 + 1 + 1} = \frac{11}{18}$$

$$P = \frac{1 + \frac{10}{12}}{1 + 1} = \frac{11}{12}$$

$$F = \frac{2 \times \frac{11}{18} \times \frac{11}{12}}{\frac{11}{18} + \frac{11}{12}} = \frac{11}{15}$$

The main issues that originates from the CEAF metric is that it is very hard to compute as it requires an optimization procedure for the evaluation. Aside of the computational drawbacks, if just a single link is forgotten that connects two system entities, then one cluster is either unmatched (and therefore does not add to the score) or is matched to a wrong entity and penalizes the score even further. This results in CEAF being the most stringent metric of those that are presented in this chapter.

BLANC

Originally promoted by Recasens and Hovy [Recasens and Hovy, 2011] and later extended by Luo et al.[Luo et al., 2014], the BLANC metric (BiLateral Assessment of Noun-phrase Coreference) is the second metric aside from MUC that is based around the links between entities. It differs from MUC in the sense, that instead of creating minimal reductions it is always creating all possible links between mentions. This circumvents the issue in MUC where big and small entities are treated equally, since the amount of links produced by a big entity scaled quadratically in its size. The second critique in MUC was its inability to incorporate singletons into the evaluation. BLANC does this by considering not only all links inside an entity (denoted **coreferential links**) but also all the links that can be created by pairing each mention from one entity with all other mentions of the other entities (**non coreferential links**). So for a singualrity no coreferential links can be computed, but the algorithm gets rewarded if it recognizes to not merge it into any other entity using the non coreferential links.

First the links between a gold entity G_i and a system entity S_j are defined as follows:

$$C_g(i) = \{(m_1, m_2) \mid m_1 \in G_i, m_2 \in G_i, m_1 \neq m_2\}$$

$$C_s(i) = \{(m_1, m_2) \mid m_1 \in S_i, m_2 \in S_i, m_1 \neq m_2\}$$

Then the sets for the non coreferential links are defined using the mentions of the entities, noting $i \neq j$:

$$N_g(i,j) = \{(m_1, m_2) \mid m_1 \in G_i, m_2 \in G_j\}$$

$$N_s(i,j) = \{(m_1, m_2) \mid m_1 \in S_i, m_2 \in S_j\}$$

The union of these sets results in the set of all coreferential and non coreferential links between gold and system entities:

$$C_g = \bigcup_i C_g(i),\ N_g = \bigcup_{i \neq j} N_g(i,j)$$

$$C_s = \bigcup_i C_s(i),\ N_s = \bigcup_{i \neq j} N_s(i,j)$$

The combination of all corefential links of the gold (system) entities and all non coreferential links of the gold (system) entities yields the set of all edges that correspond to the gold (system) clustering. Therefore defining: $T_g = C_g \cup N_g$ und $T_s = C_s \cup N_s$.

This new definition allows a comparison of the BLANC metric to the cluster rand index [Rand, 1971] which is an established metric to compare different clustering procedures. But since the amount of non coreferential links vastly exceeds the amount of coreferential links, the rand index itself would be very insensitive to changes in the set of the actual coreferential links. This is why both sets come with their own F1-score which is averaged for the final BLANC-F1. Defining the metrics for the coreferential links first:

$$R_c = \frac{|C_g \cap C_s|}{|C_g \cap C_s| + |C_g \cap N_s|}$$

$$P_c = \frac{|C_g \cap C_s|}{|C_g \cap C_s| + |C_s \cap N_g|}$$

$$F_c = \frac{2P_cR_c}{P_c + R_c}$$

Then the metrics for the non coreferential links:

$$R_n = \frac{|N_g \cap N_s|}{|N_g \cap N_s| + |N_g \cap C_s|}$$

$$P_n = \frac{|N_g \cap N_s|}{|N_g \cap N_s| + |N_s \cap C_g|}$$

$$F_n = \frac{2P_nR_n}{P_n + R_n}$$

Which are then averaged for the final F1-score:

$$BLANC = \frac{F_c + F_n}{2}$$

BLANC requires several adjustments, if the set of system mentions and gold mentions differ. It is instead suggested to use the following formulas for the calculation of the Recall and the Precision:

$$r_c = \frac{|C_g \cap C_s|}{|C_g \cap C_s| + |C_g \cap N_s| + |C_g \setminus T_s|}$$

$$p_c = \frac{|C_g \cap C_s|}{|C_g \cap C_s| + |C_s \cap N_g| + |C_s \setminus T_g|}$$

$$r_n = \frac{|N_g \cap N_s|}{|N_g \cap N_s| + |N_g \cap C_s| + |N_g \setminus T_s|}$$

$$p_n = \frac{|N_g \cap N_s|}{|N_g \cap N_s| + |N_s \cap C_g| + |N_s \setminus T_g|}$$

The formulas use the following additional sets:

- $C_g \setminus T_s$ all coreferential links that are not present in the system entities.

- $C_s \setminus T_g$ all coreferential links that are not present in the gold entities.
- $N_g \setminus T_s$ all non coreferential links that are not present in the system entities.
- $N_s \setminus T_g$ all non coreferential links that are not present in the gold entities.

If the sets of system and gold mentions are equal, then $T_g = T_s$ and the formula is reduced to the original version.

The calculation of the F scores remains unchanged.

In order to avoid a division by zero, there are three special cases that need to be respected:

1. $C_g = C_s = \emptyset$ and $N_g = N_s = \emptyset$: This occurs if there is a just a single mention inside a document. This results in: $BLANC = 1$, if the gold mentions are equal to the system mentions, otherwise $BLANC = 0$.
2. $C_g = C_s = \emptyset$ and $|N_g| + |N_s| > 0$. This means, in every gold and system entity is just a single mention. This results in $BLANC = F_n$.
3. $N_g = N_s = \emptyset$ und $|C_g| + |C_s| > 0$. The third special case is required if there is one entity in the gold clustering as well as one entity in the system clustering that contains all the mentions. In this case: $BLANC = F_c$.

Example: Revisiting the running example, given the gold entities $\{1, 2, 3, 4, 5\}$, $\{6, 7\}$ and $\{8, 9, A, B, C\}$ as well as the system entities $\{1, 2, 3, 4, 5\}$ and $\{6, 7, 8, 9, A, B, C\}$. It follows:

$$C_g(1) = \{(1,2), (1,3), (1,4), (1,5), (2,3), (2,4), (2,5), (3,4), (3,5), (4,5)\}$$

$$C_g(2) = \{(6,7)\}$$

$$C_g(3) = \{(8,9), (8,A), (8,B), (8,C), (9,A), (9,B), (9,C), (A,B), (A,C), (B,C)\}$$

The determination of the non coreferential links N_g and N_s as well as the coreferential links for the system C_s is analog. Yielding for Recall and Precision:

$$R_c = \frac{10+1+10}{10+1+10+0} = 1$$

$$P_c = \frac{10+1+10}{10+1+10+10} = \frac{21}{31}$$

$$R_n = \frac{10+25}{10+25+10} = \frac{7}{9}$$

$$P_n = \frac{35+10+25}{35+10+25+0} = 1$$

Which allows the calculation of the individual F1-scores and the final BLANC-F1:

$$F_c = \frac{2 \times 1 \times \frac{21}{31}}{1 + \frac{21}{31}} = \frac{21}{26}$$

$$F_n = \frac{2 \times \frac{7}{9} \times 1}{\frac{7}{9} + 1} = \frac{7}{8}$$

$$BLANC = \frac{\frac{21}{26} + \frac{7}{8}}{2} = \frac{175}{208}$$

The BLANC metric itself successfully got rid of the drawbacks of the MUC score. The most critized aspect is the computational power required to get the evaluation score. Assuming a document with 10000 mentions, a worst case of 10 billion non coreferential links need to be considered for every entity! For example during the evaluation of *Effi Briest* by Theodor Fontane, the evaluation of the BLANC score requires more time than the entire preprocessing, including the automatic coreference resolution. Another issue with this metric arises from the same aspect. Since it considers all non coreferent links of the solution, the so called **mention identification effect** (there are situations in which faulty detected mentions can be removed, making the system solution more precise, but causing a significant drop in the BLANC Recall) affects this metric the most.

LEA

The newest established performance metric for coreference resolution was recently proposed by Moosave et al. in 2016 and is called LEA (Link-based Entity-Aware evaluation metric). LEA is designed to be a *trustworthy* metric (the authors claim that this property holds, whenever a better system output always produces a better score, which can not be guaranteed in the previously presented metrics). At its core, LEA is an extension to the BLANC metric, it keeps the idea of evaluating all coreferential links, and models them in the so called **resolution-score** of an entity g:

$$resolution\_score(g) = \sum_{s \in S} \frac{links(g \cap s)}{links(g)}$$

The authors also found a more elegant solution to the problem to incorporate singletons into the evaluation of a link based metric. Instead of considering all non coreferential links, they added **self-links** to singletons clusters, so that each of these entities reduces into a single link. LEA does not only rely on the evaluation based on edges but also incorporates the **importance** of an entity. The authors suggest to use the amount of mentions of an entity as measure for the importance but state that this can be adjusted to different needs (e.g. one could value named entities or entities that take part in dialogues more than entities which consist solely of pronouns). Using the suggested definition for the importance, the calculation of the Recall and Precision for LEA are defined as:

$$R = \frac{\sum_{g \in G} \left(|g| \cdot \sum_{s \in S} \frac{links(g \cap s)}{links(g)} \right)}{\sum_{g \in G} |g|}$$

$$P = \frac{\sum_{s \in S} \left(|s| \cdot \sum_{g \in G} \frac{links(s \cap g)}{links(s)} \right)}{\sum_{s \in S} |s|}$$

The F1-score can be calculated as usual.

Example: Using the running example, the Precision is calculated as:

$$P = \frac{5 \cdot (\frac{4}{4}) + 7 \cdot (\frac{1}{6} + \frac{4}{6})}{5 + 7} = 0.9025$$

And the Recall can be calculated using:

$$R = \frac{5 \cdot (\frac{4}{4}) + 2 \cdot \frac{1}{1} + 5 \cdot \frac{4}{4}}{5 + 2 + 5} = 1.0$$

Even though LEA solves problems of aforementioned metrics in a more elegant way and claims to be the first metric that has aspect of both - an influence of the links and an influence term of the cluster, there are still aspects that can be criticized. The first problem is that the resolution score does already take into account that larger entities are more important than smaller ones, since the amount of edges scales quadratically with the size of the entity. Multiplying this with the amount of mentions results in a cubic weight that is assigned towards large cluster. In figure 8.2, it can be seen, that the LEA metric performs basicly identically to the BLANC metric. Furthermore none of the presented metrics really help to explain the problems of a coreference algorithm. Receiving a score as purely quantitative measure is therefore the current state of the art.

CoNLL-Score

The CoNLL score is used e.g. in the CoNLL-2012 shared task [Pradhan et al., 2012] and is calculated by averaging the three metrics: a) MUC, b) B-Cubed and c) CEAF$_e$. This is to mitigate the downsides of the individual metrics, but it is biased towards cluster based metrics, since it used two cluster based metrics and just one which is based on edges (MUC). This was exploited by Clark [Clark and Manning, 2016b], by using the B-Cube metric directly as a loss function in their system, which led to a system which was significantly better in the cluster based evaluation scenarios but much worse in the MUC evaluation score. But since two cluster based metrics are involved in the CoNLL score this was sufficient to report state of the art results.

Towards a more intuitive metric to evaluate coreference resolution

Gaining insight into the problems of an algorithm should remain one of the top priorities of a performance metric. All presented metrics fail at doing so. They allow to see if they overgenerate or if they undergenerate (due to the split in Recall and Precision). The Label-F1 score allows the creation of a confusion matrix, this matrix gives a good overview of the classes between which the algorithm has issues to distinguish, which can not be created in a similar manner using the established metrics. This section presents a sketch of a performance metric which would provide more insight into the actual problem of a coreference resolution algorithm. Most algorithms that were presented in the literature work by starting with singleton clusters and a strategy to merge them together. Such a merge can be considered as the addition of a link into the clustering. Evaluating based on links is therefore (from the algorithm's point of view) a natural procedure. The metric can then make use of good aspects of previous metrics: LEA offers the definition of an *importance* function and thus the influence of different mentions or entities can be chosen depending on the scenario. The BLANC metric introduces different F-scores that deal with different aspects of the problem. The metric that is presented here reduces the clustering to a *skeleton* of links. This concept is already used for different algorithms and is denoted as **latent tree** of a clustering. This tree is created by reducing a cluster to a series of links. But instead of creating all links or just the necessary ones, links are created based on the reading order of the text, that is if a mention of an entity has any predecessor that appears before it, then a link towards this mention is created. If no such predecessor is available, then a link to an artificial root node is introduced. This ensures, that even for singleton entities at least one edge is contained in the gold solution. The next step is a sweep in reading order of the text in order to separate the links into different semantic categories. At first the algorithm starts with an empty tree, represented by a single artificial root node. The algorithm now sweeps through the text until the first mention is found. The obvious choice is that this mention starts a new entity (this is all done to the gold clustering, I will show how to deal with the system clustering afterwards), therefore a link of the category **New_Entity** is introduced into the tree. The sweep continues until the next mention is found. This time the decision is more involved. Either the mention introduces a second entity (which would create another **New_Entity** link) or the mention is part of the first one. If the mention is part of the first entity, then a link of the category **Merge_Entity** is created and added to the tree. This process does now continue until the entire text has been processed and all links have been added to the gold tree. Since mentions usually represent *names*, *noun phrases* or *pronouns*, I propose to introduce different categories of links into the latent tree. So this would create the following categories of links:

1. **New_Entity**, if a mention is the first appearance of an entity, having three subcategories (which might be evaluated optionally): **New_Entity_Pron**, **New_Entity_NP** and **New_Entity_Name**

2. **Merge_Entity**, if a mention is merged into an existing entity. This comes with its subcategories: **Merge_Entity_Pron**, **Merge_Entity_NP** and **Merge_Entity_Name**

After the gold clustering is reduced to its links, the system clustering can be reduced based on the gold information. This would proceed as follows: The first system mention is passed, this mention is now either correct (contained in the set of gold mentions) or it is faulty. If the mention is correct, then add a true positive towards the **New_Entity** category (and/or its subtypes if necessary, for this description I will only focus on the top categories). If the mention is faulty, then the system solution adds a wrong (false positive) link for the **New_Entity** category. When proceeding through the text, some gold entities might not be present in the system solution. For each of these cases, add a false negative to the respective count, depending on whether the mention started a new entity or was combined into a previously existing one. So the only remaining cases deal with how to treat links of correct mentions. If such a mention introduces a new cluster, then it can easily be verified if this is correct using the gold standard and the according statistics are to be updated. There is only one borderline case to take care of, if according to the gold standard, the mention should have been merged into an existing entity, but **none** of the preceding mentions of this entity have been detected by the mention detection step, then no mistake is to be introduced. The last case which has not been discussed yet deals with a mention that is resolved (by the system) into an existing cluster. The question now arises, which link to consider of this active mention, as usually the system solution propagates its errors and its state might contain plenty of errors already. Due to this, reducing the link of system clusters in the same manner as the gold standard might be unwanted. For this reason I would propose to filter the system mentions of the assigned entity of the currently active mention, so that only mentions remain that appeared prior to the current mention. For this set of mentions, a partitioning with the gold clustering is performed and finally a link to the nearest mention of the biggest (in terms of mentions in the partition) entity of the partition is created. If more than one of these entities exist, then a link to the closest mention of these entities is created. This link is now either correct (according to the gold standard) or it is wrong (introducing a false positive of the according category as well as a false negative for the other category).

This process does now produce different F1-scores (one for every category) which can then be combined into a single one (if necessary, it can be weighted)

Example: Using the running example, first the gold clustering is reduced to the latent-tree (see figure 3.1, then (for the purpose of illustration only), the system clustering is reduced to the latent-tree (see figure 3.2). On two edges of the system tree, there is a mistake. The first between mention 7 and mention 8, which arises due to a missing *new* edge. The mistake between mention 8 and 9 arises, because up to this point, the system cluster consists of $\{6, 7, 8\}$ which is already flawed and represents two gold clusters: $\{6, 7\}$ and $\{8\}$. The evaluation metric assumes, that the merge is due to the larger gold cluster and implicitly maps it to $\{6, 7\}$, which results in a wrong *merge*. The correct *merge* is missing and the reason for an additional false-positive. The link between mention 9 and mention A is considered to be correct, because both corresponding gold cluster show the same size and in this case, the link is mapped to the closest mention in the clusters (which is $\{8,9\}$ and therefore correct). This results in the following scores:

$$
\begin{aligned}
P_{merge} &= \frac{8}{10} \\
R_{merge} &= \frac{8}{9} \\
P_{new} &= \frac{2}{2} \\
R_{new} &= \frac{2}{3} \\
\Rightarrow F1_{merge} &= 0.8 \\
\Rightarrow F1_{new} &= 0.838
\end{aligned}
$$

From the individual scores it can be seen, that the algorithm is better at detecting a new entity, than at merging entities.

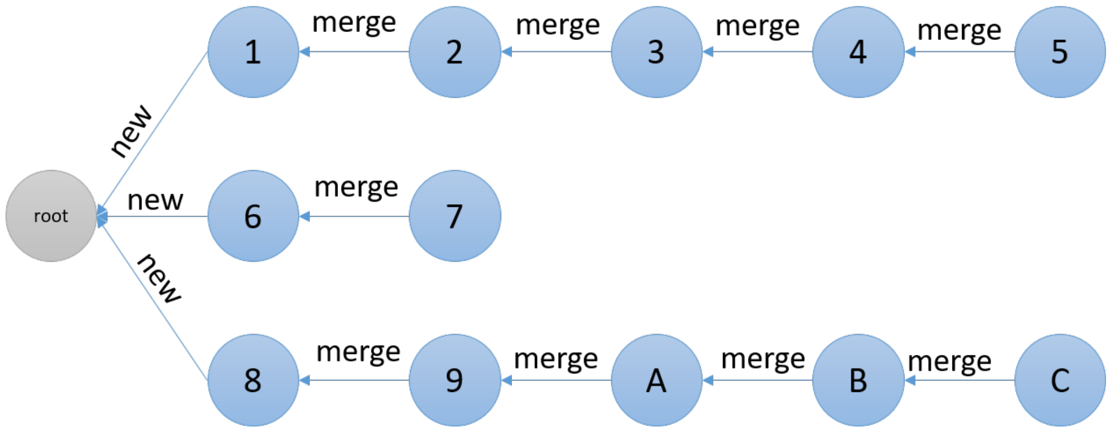


Figure 3.1.: The latent-tree of the gold clustering for the running example

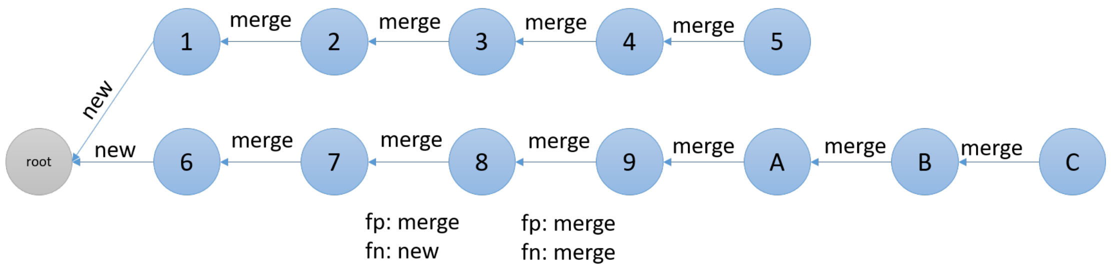


Figure 3.2.: The latent-tree of the system clustering for the running example

3.2. Background on Machine Learning

This section serves the purpose to summarize and present the algorithms based on machine learning that found its use in this thesis. Since this work comprises classical feature based, kernel based and Deep Learning techniques, I will discuss all of these fields. In the case of classical machine learning, the Perceptron algorithm, the Support Vector Machine (SVM) and the Maximum Entropy (MaxEnt) classifier were the dominant approaches for NLP tasks. Considering the Shared Task of CoNLL-2003 [Sang and De Meulder, 2003], the Maximum Entropy classifier was a part of 5 out of 11 participating systems. Even though at the core, the Perceptron algorithm is just a plain binary classifier, it can be modified to deal with multi-class, multi-label problems, integrate Kernels and can even be extended to deal with structured problems as well. The central optimization problem of the Support Vector Machine (shown in section 3.2.2) has been in the focus for decades, with a multitude of different techniques that are able to deal with this optimization problem in a clever fashion. In retrospect, the Maximum Entropy classifier is probably the most successful classifier of these three, since modern day Deep Learning algorithms usually perform their classification using a softmax operation which is followed by some sort of cross-entropy or a Loss function that uses Maximum Likelihood, which is basically a Maximum Entropy classifier. The main difference is that while a classical MaxEnt uses hand crafted features, the features are nowadays derived in an end-to-end fashion using a neural network. With most problems in NLP showing structure in one way or another, this section describes how structure can be modelled - even using nothing but binary classifier. There is an in depth explanation of Conditional Random Fields, because even though there is a lot of literature describing them, the vast majority only introduces them by mathematical means, while in section 3.2.5 I am giving an explanation about the capabilities and the flexibility of Conditional Random Fields, and how they are applied in modern Deep Learning algorithms. This section concludes with a comparison of rule-based approaches with machine learning algorithms and it highlights key aspects that even modern day machine learning approaches fail to integrate in their prediction mechanism, so that there are still clear advantages of rule-based approaches, aside from the aspect of explainability.

3.2.1. Perceptron Algorithm

The original Perceptron algorithm was proposed more than 60 years ago [Rosenblatt, 1958] but with modern modifications can still produce reliable and good results while being extremely fast. In its most used form, the algorithm takes a collection of training instances $D = (x_1, y_1), (x_2, y_2), ...(x_n, y_n)$ and learns a linear decision boundary $\hat{y} = wx + b$, with w and b being the learnable parameters of this classifier. The labels y_i for the original Perceptron can be either -1 or 1 representing two classes. it is later shown how this is extended to a multi-class classification and even towards structured prediction which is a common case in NLP.

The learning algorithm proposed by Rosenblatt, reduced to its core components is shown in figure 1.

Algorithm 1: Vanilla Perceptron Learning Algorithm

```
input : A list of training instances $(x, y)$, with $y \in \{-1, 1\}$
1  $w \leftarrow \vec{0}$;
2  foreach epoch do
3      shuffle(training instances);
4      foreach $(x_i, y_i)$ do
5          $\hat{y} \leftarrow w \cdot x_i$;
6          $L \leftarrow max(0, m - \hat{y} \cdot y_i)$
7          if $L > 0$ then
8              $w = w + \alpha \cdot y_i \cdot x$;
9          end
10     end
11 end
```

The algorithm loops through the instances for a predefined amount of epochs (a hyper parameter defined by the user), shuffles the training instances and then proceeds to loop through the individual instances. Each instance x_i is evaluated by the current decision boundary $w_t \cdot x_i + b_t$ (which is initialized by vectors of 0's in the first time step t). If it holds, that $\hat{y} > 0$, then the predicted label will be 1, else -1. Afterwards the loss function is evaluated with the predicted label. Usually the applied loss function is a simple variant of the Hinge-Loss [Crammer and Singer, 2001], being $L(y) = max(0, 1 - \hat{y} \cdot y)$. This loss is non-zero, if y and $\hat{y}$ have different labels, that is either a False-Positive ($\hat{y} = 1$ and $y = -1$) or a False-Negative occurs ($\hat{y} = -1$ an $y = 1$). If a non zero loss was evaluated, then the current weights w_t, b_t are updated using the current example x_i and a factor α, called the learning rate. There is a multitude of different ways to derive an according (adaptive) learning rate. For example, if the Perceptron should approximate weights, that classify the training example with a *margin* m, then α is set to be the Hinge-Loss $max(0, m - \hat{y} \cdot y)$. More than just a vanilla and a maximum margin variant of the Perceptron exist, an overview of them for the Perceptron is given in the very recommendable papers by Crammer and Singer [Crammer et al., 2006] and [Crammer and Singer, 2003].

This algorithm stands out, when considering modern day machine learning optimizers. If we calculate the gradient of the (basic Hinge Loss) with respect to the parameters w:

$$\vec{\nabla}(L)_w = \begin{cases} 0, & \text{if } 1 - \hat{y} \cdot y \leq 0 \\ -y \cdot x, & \text{else} \end{cases}$$

If we recall that modern Deep Learning approaches optimize using Stochastic Gradient Descent [Robbins and Monro, 1951], then the original Perceptron algorithm is just an instance of Stochastic Gradient Descent, the update mechanism originated from the Loss function, when its gradient with respect to the parameters is calculated.

With these different characteristics of update mechanisms explained, another useful feature, which is still found (in an adapted version, for example in the AdaM [Kingma and Ba, 2014] or RMS-Prop [Tieleman and Hinton, 2012] stochastic gradient descent flavours) for optimization in modern Deep Learning approaches, is the averaging of all weight vectors that were created during the optimization process. Intuitively this benefit can be expressed by revisiting the training loop. A weight vector w_t might remain stable for 80% of the train data only to be altered by a single example (which in an extreme case is even an outlier in the gold data), and there is no valid justification to completely give up on a vector that is suitable for the classification of about 80% of the data. Therefore the so called Averaged Perceptron [Collins, 2002] keeps a running sum of all weight vectors that appeared during training, scaled with the amount of time steps it remained stable. At the end of training procedure it is averaged and this vector is used for classification of unseen examples. But since even keeping a running sum is a very expensive operation compared to the regular weight update (a regular data point might comprise 50 non-zero features in NLP, while the weight vector usually has thousands or millions of features), a more clever algorithm to calculate the averaging procedure is employed, this algorithm is shown in [Daumé III, 2012].

Having a solid understanding of the binary case, the next step is to extend the Perceptron algorithm to multi-class (or even multi-label) tasks. This extension is done rather intuitively. Since a single Perceptron can classify between two classes, N Perceptrons can be applied for a multi-class classification task using N classes, where every Perceptron now decides whether the instance x is part of class k or not. Each of the Perceptrons p justify their decision using a score ($p_x = w_p \cdot x + b_p$). During prediction of unseen examples it is just the Perceptron which produces the highest score which decides the label for the example. (This approach is also called the One-Vs-All approach, however even though the Perceptrons act independently during prediction, they are not trained independently!) The only difference in the algorithm arises during the weight udpates: Each individual Perceptron predicts the example x and whenever a False-Positive or a False-Negative occurs[3], then the weights of the individual Perceptrons are updated. The training procedure can now easily intertwine the different binary Perceptrons, if the notion of a margin is introduced. Each Perceptron does now predict a score for an incoming instance x_i. If the score of the correct label is not large enough compared to all other predictions, then all weight vectors can be updated based on this information[4]. Since the MIRA update of the Perceptron was used with great success during this thesis, and I am not aware of any derivation for its update[5], I am going to present its derivation here:

Consider we got a setting in which multiple binary Perceptrons, each with their according weight vector w were used to predict an example. Since we only update as long as we made a mistake, we could adjust two of the weights. More precisely, we have to increase the score for the Perceptron which is responsible for the correct label, and the score for the Perceptron which had the highest score should be decreased, so

[3]the notations of FP and FN are different in this case, a FN occurs if the Perceptron predicts -1 for the class k the Perceptron corresponds to, and a FP occurs when the predicted label $\hat{y}$ is 1 for the current instance x, but the correct label of x is not k

[4]At its core, this change can be seen as a modification of the Loss function

[5]and the fact that I named my daughter after that algorithm - yes, my wife is aware of that.

that the next time the same training instance x_i is classified, the correct Perceptron should produce the highest score. MIRA does now introduce the following constraint on the parameter update of the Perceptrons. We denote the current weight vector which resulted in the highest score as w'_p and the current weight vector which is responsible for the current gold label as w'_g):

$$\text{minimize: } \frac{1}{2}||w_g - w'_g||^2 + \frac{1}{2}||w_p - w'_p||^2 \tag{3.1}$$

$$\text{such that: } w_g \cdot x - w_p \cdot x \geq 1 \tag{3.2}$$

In this equation, the weights after the weight update are denoted as w_p and w_g respectively. This optimization problem has to be solved for these weights. In words, this optimization problem states, update the weight vectors of both Perceptron weights as little as possible, but still enough to get a margin of 1, when classifying the same example again, so that the classification is correct. The optimization problem is solved using the same techniques as for the Maximum Entropy classifier and Support Vector Machine, by the introduction of Lagrangian Multiplier:

$$L(w_g, w_p, \alpha) = \frac{1}{2}||w_g - w'_g||^2 + \frac{1}{2}||w_p - w'_p||^2 - \alpha(w_g \cdot x - w_p \cdot x - 1) \tag{3.3}$$

Building the derivatives with respect to w_p and w_g and setting to 0:

$$\frac{\partial L}{\partial w_g} = w_g - w'_g - \alpha x = 0 \quad \Rightarrow w_g = w'_g + \alpha x$$
$$\frac{\partial L}{\partial w_p} = w_p - w'_p + \alpha x = 0 \quad \Rightarrow w_p = w'_p - \alpha x$$

This partial solution already tells us, that the update of the Perceptron is yet again a fraction of the current instance x, and we only need to determine the α for the update, this can be done by plugging the current knowledge into the constraint of the optimization problem:

$$(w'_g + \alpha x) \cdot x - (w'_p - \alpha x) \cdot x \geq 1$$

Solving for α yields:

$$\alpha \geq \frac{1 - (w'_g \cdot x - w'_p \cdot x)}{2||x||^2}$$

Since the smallest update is the update of choice, α is set to be equal to the fraction of the right hand side.

The notion of a combined training of multiple binary Perceptrons can be utilized further, in a multi-label setting (in such a setting more than one label might be considered a good prediction). Yet again, by adjusting the Loss function in a way, so that the lowest score of any correct label has to exceed a margin compared to all other scores, this constraint can be integrated into the training algorithm. This insight can be used for the prediction of a ranking in a way, that in between subsequent ranking positions, a given margin needs to be kept, otherwise the weight vector gets updated. This is very convenient if only the top rank is required and opens an entirely different perspective to the learning problem[6]. This kind of Perceptron is referred to as the **Rankceptron** in later chapters.

One last aspect reveals further beauty of the Perceptron algorithm. Recall the parameter update in the original algorithm. Whenever an update to the current weights w_t occurs, then the weights only change by (a fraction) of the vector of a training instance x_i. Even after the entire training procedure, the weight vector (which is usually initialized by just zeroes) can be seen as a linear combination of the training instances:

$$w = \sum_i \alpha_i \cdot x_i$$

After training, some instances are relevant for the creation of the weight vector w, which can be seen by an α_i that is non-zero and some instances do not contribute to the weight vector[7]. All instances that end up contributing to the final weight vector, are usually called **Support Vectors**. If we rewrite the Perceptron hypothesis using this knowledge, we get:

$$\begin{aligned}\hat{y} &= wx_j + b \\ &= (\sum_i \alpha_i \cdot x_i) \cdot x_j + b\end{aligned}$$

The multiplication with the instance x_j can be carried out in the sum[8], so that we get:

$$\hat{y} = \sum_i (\alpha_i \cdot x_i \cdot x_j) + b$$

The dot product of x_i and x_j can be carried out in a different vector space for this equation to still hold, this is commonly known as the *Kernel-trick*. To sum it up, by this simple transformation, the Perceptron algorithm can also be used with kernels at the cost of (a lot) of computational speed, because the weight vector w can no longer be recovered if the dot product requires a transformation into a different vector space, and therefore every prediction has to be made using all support vectors[9]. This derivation serves as a great base for a transition to the Support Vector Machine

[6]Instead of calculating scores for an instance and a label (which the amount is static during training), the algorithm can now create scores for a dynamic amount of candidates that get paired with an instance

[7]If the instances are shuffled in a different way, then different α values will usually be the result

[8]which is a stupid idea from a computational perspective, but mathematically no problem

[9]This is also the main reason, why any sort of kernel based classifier is incapable of solving tasks with a great number of training data.

3.2.2. Support Vector Machine

The Support Vector Machine (SVM) [Boser et al., 1992] can be considered a direct successor of the Perceptron. While the hypothesis of a SVM is equal to the one of a Perceptron (being a linear function) and even though the Perceptron already features different mechanisms to train the parameter using a maximum margin objective, there is still one distinct difference between these algorithms. The margin that is introduced by a Perceptron is only applied to a single instance (repeatedly), but a SVM introduces this margin globally with respect to **all** training instances. A SVM therefore is more of a batch version of a max margin Perceptron. At its core, the SVM solves the following optimization problem:

$$\boxed{\begin{aligned} \text{minimize :} \ & \frac{1}{2}||w||^2 \\ \text{subject to :} \ & y_i \cdot (w \cdot x_i - b) \geq 1, \forall (x_i, y_i) \in D \end{aligned}}$$

This formulation of the problem is widely known as the primal version of the SVM. There are approaches that will directly solve this version of the SVM (namely e.g. Pegasos [Shalev-Shwartz et al., 2011]), but it is most common to solve the dual version of that problem [Yu and Kim, 2012]. The solution of this different view of the same problem is (in this case at least) also a solution to the primal. The dual formulation of the SVM states:

$$\boxed{\begin{aligned} \text{maximize:} & \sum_i \alpha_i - \frac{1}{2}\sum_i \sum_j \alpha_i \alpha_j y_i y_j x_i x_j \\ \text{subject to:} & \sum_i \alpha_i y_i = 0 \\ & \alpha \geq 0 \end{aligned}}$$

The solution of this problem is now denoted by Lagrangian multipliers α, which denote the influence of individual instances x into the creation of the separating hyperplane. Even though many approaches for the solution are available, the most prominent algorithm for the solution of this optimization problem is yet again of iterative nature and called the *Sequential minimal optimization* algorithm [Platt, 1998]. This algorithm solves the optimization problem by always considering two weights α_i and α_j and updating both of them, while respecting all constraints. After a number of iterations, the weights will converge to the optimal solution. A version of pseudo code of the SMO algorithm is shown in [Lozano-Perez, 2001].

If an SVM (which is binary by nature) is to be applied to more than just two classes, then a multitude of SVMs are trained, either in a One-vs-All fashion or in a One-vs-One fashion. A comparison of these can be found in [Chang and Lin, 2011].

3.2.3. Maximum Entropy Classifier

In this section I will finally introduce the Maximum Entropy classifier. If you are coming with a background of Deep Learning, this might appear to be a strange name for a Softmax with a Maximum Likelihood Loss. Originally it was introduced in the context of probabilistic graphical models and can be considered as a successor to the Naive Bayes classifier [Friedman et al., 1997]. The main idea is based on the principle of maximum entropy by Jaynes [Jaynes, 1957]:

> If incomplete information about a probability distribution is available, the only unbiased assumption that can be made is a distribution which is as uniform as possible, given the available information.

This is reflected in the conditional entropy, which is maximized, if an uniform distribution is utilized, so the goal is to find a model $p^*(y|x)$ that maximizes the conditional entropy $H(x)$:

$$H(y|x) = -\sum_{(x,y)\in D} p(y,x) \cdot \log p(y|x)$$

while being consistent to the training data D. A model is considered to be consistent to the training data if the constraints introduced by *feature functions* are satisfied. A feature function $f_i(x,y)$ is an indicator (usually a binary), whether a given expression holds true in the data. In general:

$$f_i(x,y) = \begin{cases} 1, & \text{if a boolean expression holds true} \\ 0, & \text{else} \end{cases}$$

The amount of feature functions are to be defined by the user, as long as they are of the aforementioned form, then even millions of features pose no problem to the classifier. These features act as constrains to the model. Consider the empirical expected value of any feature function f_i in the training data:

$$\hat{E}(f_i) = \frac{1}{N} \sum_{(x,y)\in D} f_i(x,y)$$

In reality this is just a vector with the normalized counts of the features in the training data. Any model is considered to be valid (all valid models are to be contained in a set P) if the expected value of the feature functions $E(f_i)$ is equal to this vector of relative counts. The expectation of the model features is calculated as follows (derivation omitted, see [Klinger and Tomanek, 2007]):

$$E(f_i) = \frac{1}{N} \sum_{x\in D} \sum_{y\in Y} p(y|x) \cdot f_i(x,y)$$

With Y being the label set. In total, this results in the following optimization problem:

$$
\begin{aligned}
p^*(y|x) = \max_{p(y|x) \in P} H(y|x) \\
\text{subject to:} \\
\hat{E}(f_i) - E(f_i) = 0,\ \forall f_i \\
1 - \sum_{y \in Y} p(y|x) = 0,\ \forall x \\
p(y|x) \geq 0,\ \forall x, y
\end{aligned}
$$

The first condition arises from the feature functions, and the second and third condition only ensure that the sum of the probability of the events equals to 1 and every probability is greater or equal to 0 (so that the result is a valid probability distribution). This optimization problem can be solved using the method of Lagrangian multiplier [Rockafellar, 1993], and the solution is as follows[10]:

$$
p(y|x) = \frac{\exp \sum_{i=1} \lambda_i \cdot f_i(x, y)}{\sum_{y' \in Y} (\exp \sum_{i=1} \lambda_i \cdot f_i(x, y'))} \tag{3.4}
$$

The Lagrangian parameters λ_i that were introduced during the optimization are to be fitted during training and are considered to be the parameters of this model. After the optimization, the vector of predicted counts of the features should be equal to that of the empirical counts, so the model basically selects parameters, so that the features can be replicated in the training data. Training a MaxEnt model is done by gradient based optimization methods, usually in a batch fashion using optimizers such as LBFGS [Nocedal, 1980], Conjugate Gradient [Hestenes and Stiefel, 1952] or Orthantwise LBFGS [Andrew and Gao, 2007]. Originally dedicated optimizers for the training of this classifier were released and are known under the name of Generalized Iterative Scaling (GIS) [Darroch and Ratcliff, 1972] or Improved Iterative Scaling (IIS) [Berger, 1997]. The MaxEnt classifier can also deal with structured input, the classifier is then known under the name of Conditional Random Fields. Section 3.2.5 introduces CRFs and explains how they are used in Deep Learning applications. In order to understand this presentation, a short introduction of Deep Learning is given in the next section.

3.2.4. A primer on Deep Learning

The purpose of this section is not to provide an in depth introduction of Deep Learning, it just outlines the differences compared to classical machine learning, so that CRFs which are presented in the next section can be understood in a more general way. For a more in depth introduction of Deep Learning, refer to [Goldberg, 2017] or

[10]In the original publication of Berger, he refers to the solution of this optimization problem as "simple calculus", which is funny considering the length of its derivation comprises more than a single page of derivation steps

[Goodfellow et al., 2016]. The main difference is the form of the hypothesis that is modelled by the classifier. While the previous three methods only had linear or log-linear hypotheses, Deep Learning is capable of representing a large variety of different functions. In general, the hypothesis $H(x)$ is a composite function (in fact arbitrary many functions can be included):

$$H(x) = f_{\theta_f}(g_{\theta_g}(h_{\theta_h}(j_{\theta_j}(x))))$$

Each of the functions may have their own parameters θ, which are to be learned during training. If the focus is on the understanding of a Deep Learning framework, then such a function, should be modelled as an elementary operation that is executed by the computer. An example for an elementary operation is an addition, a matrix multiplication or a convolution. In order to allow an automatic differentiation, and with that a general procedure to train those arbitrarily interleaved functions, for each of these elementary operations, a **forward** and a **backward** pass needs to be provided (by the developer). The forward pass evaluates the operation *op*, when an input is given, and the backward pass calculates the derivation of *op* to its inputs. Consider the example for the matrix multiplication:

Matrix multiplication operation (input matrices X and W):

$$forward : Y = X \cdot W$$
$$backward_X : \frac{\partial L}{\partial X} = \frac{\partial L}{\partial Y} \cdot W^T$$
$$backward_W : \frac{\partial L}{\partial W} = X^T \cdot \frac{\partial L}{\partial Y}$$

If the outermost operation is a function which produces a scalar value (which is the case with loss functions), then the required derivations for a gradient based optimization can be determined automatically using a variant of the backpropagation algorithm [Rumelhart et al., 1988] which is called Reverse Mode Automatic Differentation [Baydin et al., 2018]. Reducing Deep Learning to a set of elementary operations is crucial if a flexible framework is to be built. Consider e.g. Recurrent Neural Networks (such as GRU [Cho et al., 2014], LSTM [Hochreiter and Schmidhuber, 1997] or RHN [Zilly et al., 2017]). Talking in layers is fine when communicating with fellow researcher, but at its core a layer is just syntactic sugar that adds a bunch of operations (dynamically many in the case of a recurrent layer) onto a computation graph which consists of elementary operations. Such a computation graph can be represented as an arbitrary object representation and be executed (alongside all its inputs, which is the data and the parameters) on a GPU in order to obtain a huge speedup. This principle of a dynamic amount of elementary operations is crucial for the understanding of neural CRFs.

3.2.5. (Neural) Conditional Random Fields

This section introduces CRFs and shows how they are used in modern Deep Learning approaches. There are many papers that used CRFs but the amount of easy to grasp sources is scarce[11]. I recommend to read [Klinger and Tomanek, 2007] and [Collins, 2015] for an overview of CRFs. CRFs are more than just a single formula. Instead they form an entire framework, where adaptations to new tasks or new structures can be made by using different *templates.* A template can basically be considered as its own classifier, comprising its very own weights.

The purpose of CRFs is to help the classifier make sense of a structural output. A very frequent structure is that of a *sequence*, as can be seen in chapter 6. For the purpose of this section, we will only discuss a sequential Part-of-Speech tagging task with three (positive) labels **V, N, AD**, being one class for verbs, one for nouns and one for adjectives (additionally there is label **O** for the class other). The task is to predict a sequence of these labels for an incoming sequence of tokens. This problem could be handled by simply predicting one label for each of the incoming tokens (using features of choice) using e.g. a MaxEnt classifier. This is a valid approach, however it completely ignores information in the *label sequence* (e.g. it is more unlikely to have two verbs follow each other than it is to have a noun follow an adjective). In this naive approach (using the MaxEnt classifier), the formula 3.4 would be applied independently at every incoming position. Features could still be calculated from the entire sequence, but information about previous decisions are ignored. If the MaxEnt classifier is applied at a position t in the sequence, the label that gets chosen has the highest *score* (I am referring to the numerator of the MaxEnt as scores) at every position. Using this, the score (ignoring the denominator) for the entire sequence X with label sequence Y:

$$score(Y|X) = \sum_{t \in sequence} score(X, Y_t)$$

In order to evaluate this equation, we have to provide a label sequence Y and a sequence of inputs X, and we get a score for the compatibility of Y to the sequence X. We can now simply build all possible sequences and evaluate the score for them, the sequence which gets assigned the highest score is the one to chose for an input. The problem is, that the amount of all sequences is of an exponential nature, exceeding even modern day computation power relatively quickly. In the example above, since we calculated the score for a sequence *independently* for every position, we do not need to consider an exponential amount (but merely:[amount Labels $\cdot$ sequence length] many). This evaluation is due to the way we calculated the scores for the sequence. If the score for position t is calculated using more than just the label Y_t, then the computation becomes more expensive. Calculating scores for sequences in a more complicated way is still possible without loosing tractability [12]. If the score at position t only considers information about previous positions in the sequence, then (by the usage of dynamic programming) the best sequence can be determined in polynomial time.

[11] At least, in my opinion

[12] As a matter of fact, there are approximate ways to calculate even complicated scores, using Junction Trees [Barber, 2003], Loopy Belief Propagation [Kschischang et al., 2001] or Metropolis-Hastings sampling methods [Sutton, 2008]

Using this intuition, we can further introduce a template Φ. As previously stated, templates are at the core of CRFs as they determine how to calculate the score for a given structure (in this case a sequence). In general, for a sequence, the score of it can be calculated as:

$$score(Y|X) = \sum_{t \in sequence} \sum_{\Phi \in templates} score_{\Phi}(X, Y)$$

This equation means, that the score of all available templates is summed over the steps in the sequence in order to calculate the score of the sequence. In the aforementioned case, there was only a single template, which calculated a score based on X and Y_t. Since this template does not need any information about any other labels in the sequence, this template is named *Node-template.* The most common other template is the *Order-1-template.* This template calculates the score based on the input X, the label at the current position t and the previous position $t-1$. The score for a sequence would change to:

$$score(Y|X) = \sum_{t \in sequence} \textit{Node-template}(X, Y_t) + \textit{Order-1-template}(X, Y_t, Y_{t-1})$$

While the Node-template provides information about how good a label Y_t fits to a current token X_t (using all sorts of features in X), the Order-1-template provides information about how good a label Y_t fits, if we observe a previous label Y_{t-1}. Both templates contain their own set of features and weights. These two templates together are used in a so called Linear Chain CRF, which is the most common use case of CRFs[13]. In order to get a deeper understanding of CRFs, I am providing an example as of how the score of a sequence can be calculated efficiently. The example sentence is "The old man eats fish", with a correct label sequence of "O AD N V N". Scores for a sequence are summed up in a *lattice-diagram.* At every position in the sequence, the lattice contains a node for every possible label. At the beginning, there is a dummy start label \$. Between each label of a given time step, all valid transitions to states of the next time step are represented by edges in the lattice. The edges are scored using the templates Φ. At every node, out of all incoming edges, only the edge with the **maximum** value, consisting of the sum of the scores of the previous node and the sum of the scores provided by the templates for an edge is retained. On top of this score, the score of all templates that operate on just the node level is added (usually this is just one template) and this sum is stored at that node alongside the information from where the best incoming edge was coming. This algorithm is widely known as the Viterbi algorithm [Forney, 1973]. At the end of the lattice a special end node <*end*> contains the maximum score of all sequences and the sequence itself can be recovered by following the backpointers, see figure 3.3.

[13]In this setting, the features of the Node-template are often referred to as back-off features.

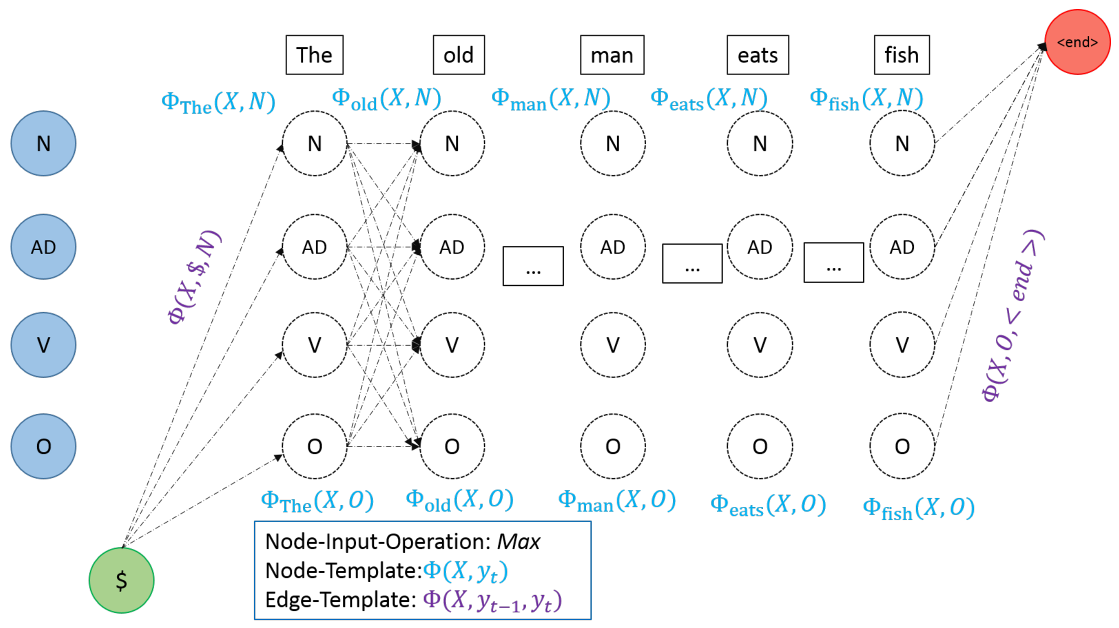


Figure 3.3.: Lattice diagram with a selected amount of edges and the templates that are assigned to the nodes and the edges. It starts at a dummy start node \$ and ends at a dedicated end node.

Seeing this lattice diagram, one can grasp, that the score for a sequence is summed using the edges and the nodes alongside their templates. But this understanding will not suffice in order to understand the concept of more complex CRFs, for example what if an additional template should be utilized which calculates the score based on two previous states $(X, Y_t, Y_{t-1}, Y_{t-2})$, known as the Order-2-template. In the previous lattice, it is not clear at which edge the template should be evaluated. This is why the lattice itself is not at the core of a CRF instead a more general **transducer** which can generate these lattices. All which is required for the Viterbi algorithm is an amount of states, a set of valid transitions between the states, the length of the sequence and - for each transition between the states - a set of templates that are assigned to it. The transducer for the current example is depicted in figure 3.4. If we add the Order-2-template, then the states of the transducer and the transitions of the transducer will change (into tuples of states), so that all templates can now be assigned to the edges (the Order-1-templates are now assigned to more than one transition, but it is up to the implementation, how to deal with that).

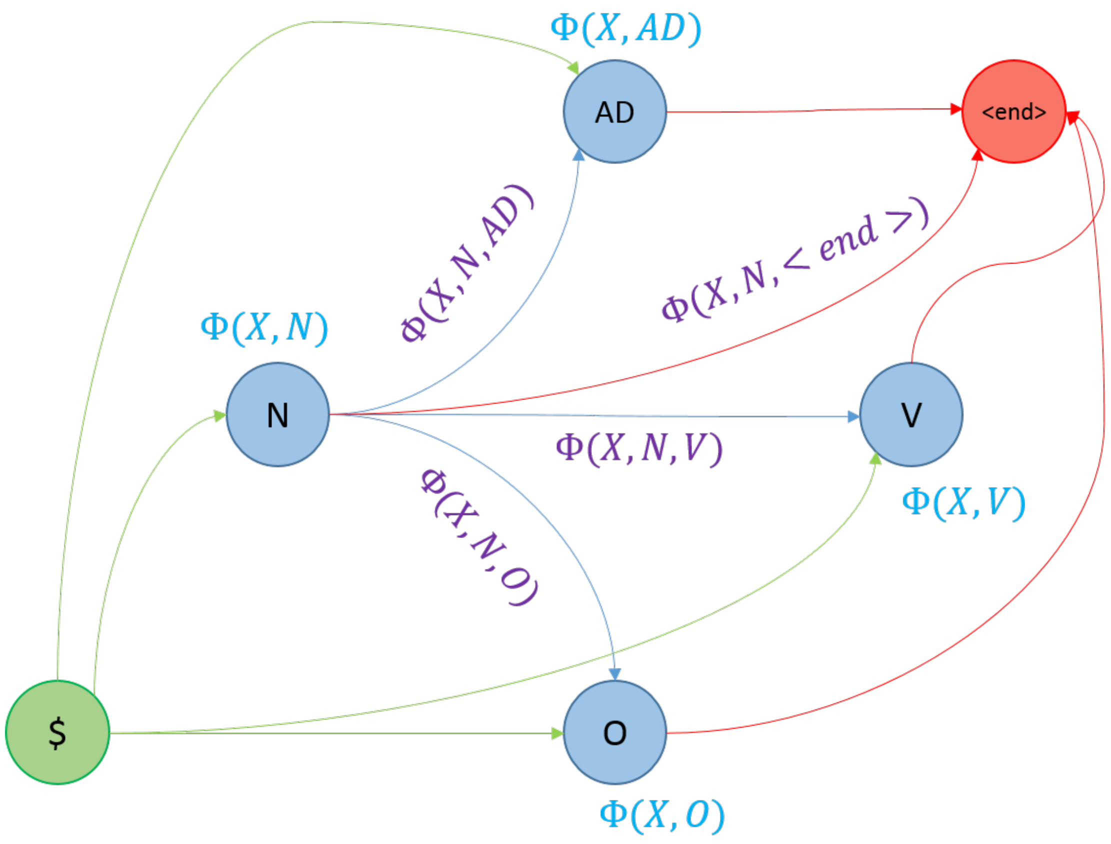


Figure 3.4.: Transducer for the running problem, with a selected amount of edges and templates assigned to it.

This transducer can now be passed to the Viterbi algorithm which produces the best sequence using the available states and transitions. So to provide a concept for CRFs:

1. Define a set of templates Φ
2. Build a transducer for the current set of labels Y and add all transitions according to the templates Φ (and the available training data)
3. Decode a sequence (using the current weights) using this transducer and the Viterbi algorithm.

In order to understand the training of the CRFs, a more general form (for sequences) of them can be formulated as:

$$p_{CRF}(Y|X) = \frac{\sum_{t \in sequence} \sum_{\Phi \in templates} score_{\Phi}(X,Y)}{\sum_{Y'} \sum_{t \in sequence} \sum_{\Phi \in templates} score_{\Phi}(X,Y')}$$

The numerator calculates the score for a current assignment Y and the denominator is the sum of all scores of all sequences, and therefore normalizes a score to a probability. The numerator is calculated using the Viterbi algorithm and the denominator is calculated using the **Forward** algorithm [Rabiner, 1989][14] The parameters (which are all stored in the templates) are usually initialized to 0 and a CRF is trained using gradient-based optimization and a Maximum Likelihood Loss function, so it tries to assign a probability of 100% to all valid sequences that are observed during training. I will omit the gradient of a CRF, instead the interested reader can find the derivation in [Klinger and Tomanek, 2007]. Having a system which is capable of building the gradient automatically (presented in section 3.2.4) one can verify, that all the Forward algorithm and the Viterbi algorithm do is to add operations into a computation graph of which the derivation can be formed automatically. Similar to Recurrent Networks, the amount of operations that is required changes for every sequence (depending on the templates and especially the length of the sequence).[15]

Summing up regular Conditional Random Fields, before stepping into their neural counterpart. Using CRFs is thinking in **templates**, which is the main building block in order to adapt them to new problems. A few prominent examples and their use:

- Order-k-template: This is the most common template, which calculates the scores based on the current label Y_t and k previous labels. It is the default for sequence classification. Most of the time an order of one or two [Sha and Pereira, 2003] are used.
- Semi-template: This template counts how many times the same label is used after another. A transition in the transducer of this template might be $(N, 4) \rightarrow (N, 5)$ showing that from a state which had four nouns in a row a new state which has five is reached [Sarawagi and Cohen, 2005]. It is a common choice for reference segmentation or it could be applied to the detection of direct speech, since the system is capable of evaluating the use of a feature given the current length of a speech.
- Multi-task-template: This template can be used to utilize constraints in between different (sequential) tasks. Its use was demonstrated by the combination of POS-tagging and chunking [Sutton et al., 2007]. A state in this setting might contain $(N, I - VP)$ showing that the current POS-tag is a noun and it is in a verb phrase (which is highly unlikely, but it demonstrates how constraints between different tasks can be utilized in this manner)

Using CRFs in applications with Deep Learning is currently considered as state of the art for sequential prediction. This section does (briefly) explain, how the framework of CRFs is adapted to be used in a neural network. A CRF can be thought of as a layer

[14]The forward algorithm is almost equal to the Viterbi algorithm, but instead of taking only the maximum score at each node, the forward algorithm uses the sum, since it is meant to capture the sum of all paths.

[15]You need a very good automatic differentiation system, since the calculation involves sums of exponential functions, which might end up causing numeric problems.

(even though we know that a layer itself is just syntactic sugar) that can be used instead of a softmax layer. A softmax creates a probability for every label in the labelset, this is not feasible for a CRF, since the amount of labels (which are all different sequences) is exponential in nature. This is why a CRF layer usually incorporates the Maximum Likelihood Loss in the layer itself. A further distinction to a softmax layer is that a CRF layer contains additional weights for the network, the weights originate from the different templates which form the CRF. I will only go over the simplest form of a Neural CRF, since this is what is used in most application which talk about using *Neural CRFs* for sequence prediction. This form is an order-1 linear chain CRF. This means, that there are two templates, the first being the Node-template and the second template is the Order-1 template. The features for the Node-template are identical to those that were usually fed into the softmax layer (so they are just the neurons of the final layer, usually called logits). The edge scores for the other template is realized by storing an additional matrix of weights in the CRF layer. This matrix is of the shape $L \times L$ with L being the amount of different labels in the labelset. So there is a weight which gets added for every transition in the lattice. The scores are summed up in an equal way as in the classical CRF, adding operations into the computation graph using the Viterbi and the Forward algorithm for the numerator and denominator respectively.[16]. Obviously by modelling different inputs and templates into the neural CRF layer, additional edge scores can be added. This however comes at the cost of modelling it using a transducer, which is currently not supported in modern Deep Learning frameworks and needs to be added by yourself.

3.3. Rule-Based Algorithms

This section introduces the general approach that was taken, whenever a rule-based approach was developed in the context of this thesis. Similar to machine learning algorithms, a *rule-development* data set needs to be available. The purpose of this distinct (from the final test set) data set is to verify the implementation and to develop and extend the existing rule-system. Throughout this work, all algorithms start with a general idea of a procedure, many with the question in mind as of how *"would I solve this problem"*. In general, this does already provide an outline of an algorithm. I am going to illustrate this with the example of a rule-based character reference detection. The relevant steps for the general outline of an algorithm are as follows:

1. A character can be recognized by the token itself.
2. A character can be recognized in the context of several verbs or adjectives.
3. A character is a reappearing entity of the text.
4. A character can be represented using many components, such as titles, first name or last name.

[16]The implementation can be seen for Pytorch: `https://pytorch.org/tutorials/beginner/nlp/advanced_tutorial.html`, accessed 20.05.2020

Even though such a general procedure can be written down, it does not mean, that it is easy to implement by any means.

Recognizing a character by the token itself is a task, that can be modelled using gazetteers, so the quality at this step does greatly depend on the resources that are available. In this scenario, having more resources or larger gazetteers is no promise for better results, due to the vast ambiguity of language and more resources might mix up the aspects of different domains too much. The second aspect centers around the local context of a token. It is apparent that some verbs (especially verbs reflecting communication) have a high chance of being used along a fictional character, the same applies for some adjectives (e.g. *beautiful*). An implementation of this aspect is already problematic, since the resources that are required for this need to have a combination of frame information and semantic information from a taxonomy (so the best case would be a combination of GermaNet and Salsa, the German version of FrameNet, which at the time of writing is not available), so that e.g. it can be inferred that the agent is either an *organization* or a *person*. Furthermore an implementation of this aspect relies on the detection of subjects and objects, which in itself is by no means a solved task in the NLP-community, especially when there is a shift in the domain. The third aspect can be implemented rather easy compared to the previous aspect of the local context. Not only does it allow to improve upon missed instances that could not be detected by any of the previous aspects, but this rule tends to be more important, the longer the texts in focus are. The last aspect seems to reuse the local context of a token again, however this is not necessarily true. If one examines all local contexts of a capitalized (this does only hold for German) token, then one can yet again utilize a global view on text instead of just using the local aspect. A character can then be recognized using a restricted number of rule templates (e.g. "Title","Firstname", "Capital Token"), where the last capital token can be recognized as a last name with high confidence.

During this first phase of the creation one can not only estimate as of how long the implementation would take (and which aspects are hard to implement) but also already determine the interfaces where such an algorithm can be adapted (e.g. by improving the resources). The second phase applies the initial implementation to the development data and usually this reveals situations that were forgotten by the rule engineer (An example for this is that a sequence of capitalized tokens, where one token is in a genitive position should be treated different, see *"Angela Merkel" vs "Angelas Hand"*). In this phase one can integrate some aspects of the domain into the algorithm, this is escpecially useful, when rules show conflicts in their indication (and I have never seen a scenario in which they wont). Resolving a state of conflicting rules is not easy, and most of the time it is reduced to a scoring mechanism that is introduced, with the scores for a rule being set by intuition or measured using the available data. This second phase has clear potential to overfit the algorithm to the given domain and at this point in time, I cannot give a way to prevent this. However one can always compare the final results on the training and the test data to uncover if this actually happened (throughout this thesis I had worse results on the training data compared to the test data, showing that this approach does not necessarily lead to an overfitting).

When, no procedure can be found that utilizes different aspects of the data (e.g. local features vs global features), and the cost for labeling data is cheap, then a creation of a rule-based algorithm might not be a good choice. The next section goes over the aspects as of when a rule-based algorithm might still perform on par with modern day machine learning and in which situations it might not.

3.4. Comparison of Rule-based vs Machine Learning Methods

This section concludes the review of different aspects of machine learning and compares their capabilities with those of classical rule-based approaches. Its main purpose is to justify the existence of rule-based approaches by utilizing the limits of current machine learning techniques. I am trying to give an outline of the circumstances that need to be fulfilled, if a rule-based system is to be chosen in favor of a machine learning system. This is done by presenting four weak spots of machine learning algorithms that can be exploited by a rule-based system, in order to achieve competitive results:

Machine Learning is based around a static Scenario: In almost all applications, the instances which are fed into a classifier follow a static scenario, that is they all depict e.g. single tokens, or single sentences (the latter is the more common case). Consider the task of NER, which is a classical sequence prediction task. The truth is, that this is not entirely true. It usually is sufficient to detect a name once in a document and this name can be spread to the text in the entire document. A pure sequence classifier will reattempt to classify the same name over and over again. A post processing step which could accomplish the same is highly problematic as well since the classifier is not trained so that its decisions can simply be used to spread (e.g. by considering the confidence) through the entire document (also it is rather unclear how to deal with B and I labels, since this poses the risk to merge adjacent entities). A rule-based system can apply its high Precision rules and then distribute the results to the entire document. Even better, the rule-based system can take the results of the high Precision rules, then generate its most likely morphological forms (e.g. if they would appear in a genitive position) and then match this resulting set to the text. In this manner, the rule-based system can change the scenario of its instances in a very flexible manner and exploit constraints a machine learning system is not capable of.

Machine Learning relies on Training Data: Probably the most severe bottleneck for the application of the machine learning algorithm is that it needs training data. An example that made me struggle for quite a bit throughout this thesis are syntactical parser. There are only German models that are trained on newspaper corpora and none for German literary texts. This results in the parser being rather unreliable since it is not trained to be used in a different domain. While these drops in quality are also expected for a rule-based system, it usually is better to adapt a rule-based system (if the rules are documented somewhere) than to reannotate an entire data set for the new domain (even though I feel quite secure with the syntactical functions of the German grammar, annotating a data set for a syntactical parser is still a really challenging task).

Machine Learning usually is applied for a single task only: This is another stumbling block, especially when sticking together a multitude of machine learning algorithms in a pipeline. The systems, even though they claim to have state of the art performance scores, are just not compatible in a sense, that mistakes that are introduced by a previous system confuses a later system. Rule-based systems on the other hand can adapt to the input, as long as it is generated by a rule-based system itself and the results can be comprehended by the rule engineer.

Data for Machine Learning only shows Correlation: Rule-based systems deal with causality in its most natural way. An *if-then* clause is just what can be considered as causality. Data for Machine learning is usually represented in a tabular form and does not contain any causalities whatsoever. Using a tabular input, there are currently only methods to detect correlations between features but nothing more.

Even though the aforementioned paragraphs make it sound as if machine learning still has some serious drawbacks, there is a lot of ongoing research (and existing techniques) to mitigate these problems. Current Deep Learning approaches can for example use pretraining in order to reduce the amount of training data that is required, both for adapting to a new domain or to a new task. If possible, Deep Learning aims to treat a task in an end-to-end fashion, so mistakes introduced in a pipeline can no longer confuse downstream algorithms. There is a lot of ongoing research to integrate prior knowledge into Deep Learning, this field, is known as Bayesian Deep Learning, and is still rather new [Shridhar et al., 2019]. The issue which is built around a static scenario of machine learning originates from its past. Not even 20 years ago, training a SVM with a data set of reasonable size was a real challenge, with more computational power, more complex models can be built, that are able to learn on larger scopes (e.g. take an entire document as input and predict the names within). So this list is only able to give a current snapshot of the state of machine learning versus rule-based approaches and if I would rewrite this section at a later point in time, circumstances may have changed and result in an entirely different section.

3.5. The Convergence of Rule-based approaches and Machine Learning algorithms

This section is highly subjective, but I want to speculate about the state of these two aspects in the future. With Deep Learning getting *"replaced"* by *"Differential Programming"* (which is in essence the same technique), the difference of a rule-based algorithm and a machine learning algorithm might vanish over time. The implementation of each aspect (e.g. the aspects as shown in section 3.3 for the character reference detection) is in essence a snippet of code, consisting of classical programming control flow (loops and branches). All this leads to a data structure in the code (the body of the rule), and this data structure can also be represented using a neural representation. For each of these data structures a scoring mechanism can be easily incorporated by a feed-forward architecture leading to a single node, and a comparison of these scores can be done using a maximum operation, or a neural sorting algorithm, if a ranking is of interest. So in the future, a rule-based program might use a neural architecture to improve upon the weak points of a rule-based system (e.g. evaluating a context, or scoring a rule), while the general flow of the program resembles a rule-based algorithm. The requirement for a such a mix is a programming language that supports an inherent differentiation in all its operations and the compiler of Swift[17] is currently reworked to support just this. So my guess is that in the future, the control flow of rule-based systems will remain, but the *endless aspect* (such as scoring each context of a token) is taken care of by a neural network. So in my guess it will simply be a shift of programming style, that will eventually lead to a convergence of a rule-based algorithm which is trained using gradient descent.

[17]`https://www.apple.com/de/swift/`, accessed 20.05.2020

4. Datasets and Resources used in this Work

[1]This chapter elaborates the data sets as well as the resources (either external or resources that were created during this work) that were used in experiments throughout this work. It starts with a review of available resources for the given tasks and goes into detail for the creation of new corpora specific to the literary domain.

4.1. Available Corpora

This section serves as the related work for corpora available prior to this work for the tasks of Named Entity Recognition, Coreference Resolution as well as Speaker Detection.

4.1.1. Datasets for German Character Reference Detection

Since character references are in general just a set of special noun phrases, the task can be best compared to that of Named Entity Recognition, where only the noun phrases are of interest that form names. Aside from DROC, there are two well known datasets for NER. The first was released as a part of the CoNLL 2003 shared task [Sang and De Meulder, 2003]. It consists of about 300.000 tokens of annotated newspaper articles and contains four different labels: locations(LOC), organizations (ORG), persons (PER) and the remaining entities are labelled as miscellaneous (MISC). The second dataset was released during the shared task of the GermEval 2014 [Benikova et al., 2014] and comprises about 600k tokens. It has the same four labels as the CoNLL dataset, but the guidelines differ between outer and inner spans. For example *Bayern München* is tagged as an organization, while both *Bayern* and *München* are locations. Further distincions to the schema of CoNLL are: a) there are derivations of tags (e.g. in *Berliner Journalisten*, the token *Berliner* is tagged as LOC_{deriv}), b) parts of compounds are also labelled (e.g. in *Hamiltonoperator*, the compound *Hamilton* is labelled as PER_{part}) c) The dataset is a set of different sentences, compared to a set or complete articles which is the case of the CoNLL dataset.

There are datasets based on historical texts [Neudecker, 2016], that are extracted from the Europeana collection of historical newspapers[2]. There are two distinct sets for German data, the first contains newspaper pages of Dr. Friedrich Teßmann library in South-Tyrol (about 87.000 tokens) and the second one pages from the Austrian National Library (about 35.000 tokens). The texts are written in 19th century Austrian German

[1]The description of DROC was published in [Krug et al., 2018a]

[2]`https://www.europeana.eu/portal/de`, accessed 20.05.2020

and can be considered vastly different to common modern day language. Similar to the CoNLL guidelines, these two corpora contain the same four labelled categories of entities. This means, that at the time of writing, DROC is unique because it contains not only pronominal noun phrases as names, but also appellatives that refer to characters. On the other hand, DROC has only a markup for characters and no annotations for locations or other types of entities.

4.1.2. Datasets for Quotation Attribution

At the point of writing, aside from DROC, no other resources for German quotation attribution are available. There is one resource, the Columbia QSA corpus [Elson et al., 2010], which consists of fractions of 16 english novels in which quotations and their according speaker have been labelled manually. The final corpus comprises 3176 annotated quotes for which speaker were available. Their annotation differs in one crucial aspect, when compared to DROC: Pronouns are not marked as speaker, but instead, the closest reference of the according entity is marked as the speaker. This decision was made, because no gold coreference was available. This corpus was extended by [Muzny et al., 2017] into the QuoteLi3 corpus, this dataset covers three entire novels with in total about 3.000 quoted speeches and their speaker. It is composed of dialogue from Jane Austen's *Pride and Prejudice*, *Emma*, and Anton Chekhov's *The Steppe*.

4.1.3. Datasets for Coreference Resolution

Comparing the DROC corpus in the field of coreference resolution yields a number of similar resources – though none in the domain of (German) literary texts. This section is restricted to German and English corpora available for academic research, starting with the latter. The best known corpora for English were released in the scope of the MUC-6 and MUC-7 conferences ([Grishman and Sundheim, 1996] and [Chinchor and Robinson, 1997]). These corpora each comprise about 30.000 tokens and contain articles from the Wall Street Journal (WSJ) and airplane crashes. The corpus released for the ACE conference of the year 2005 ([Walker et al., 2006] had about 400.000 annotated tokens and contains a mix of news, blog and web articles. With about 1.500.000 tokens, OntoNotes 5.0 ([Weischedel et al., 2013]) currently is the largest available resource for coreference resolution and consists of news articles, conversations and web articles. For German, there are currently two corpora available. The first is the Potsdam commentary corpus ([Stede, 2004]), comprising 33.000 tokens derived from 176 newspaper commentaries. The other, far larger resource for German coreference resolution is the TüBa-D/Z corpus, released by the university of Tübingen ([Telljohann et al., 2006]). It is made of about 3.400 newspaper articles, with about 1.500.000 tokens. This overview shows that there is currently no resource for (German) literary texts and most articles of the aforementioned resources tend to be much shorter than an average novel – yielding new phenomena to explain with statistical methods, therefore justifying the release of DROC, a resource comprising 90 fragments of German novels, published between 1650 and 1950.

4.2. The Creation of DROC

[3] DROC (**D**eutsches **RO**man**C**orpus) is the main textual resource that was manually annotated during the Kallimachos project, in its current state it comprises 90 fragments of novels with close to 400.000 tokens. This section describes the textual resources, the annotation guidelines, the resulting Inter-Annotator Agreement and some statistics prevalent in this corpus.

4.2.1. Description of the Textual Sources

The texts of the novels which are the basis for our corpus come from a large collection of German literary texts available as full-texts, part of the TextGrid repository[4]. The texts found in this repository are part of one of the first large-scale digitization projects in the German language. The digitization was undertaken in separate steps by a commercial company, Directmedia, over the course of ten years, which sold digital texts on CDs and DVDs. It is important to understand that the TextGrid collection comprises two different groups of texts: The first group, by far the largest, consists of canonized texts of German literature. These are usually based on scholarly editions used for decades by academics. In most editions the writing has been normalized: In our context this means mainly that ”th” has been replaced by ”t” (for example ”Tür” instead of ”Thür”) and ”ey” by ”ei” (for example ”sei” instead of ”sey”). The second group has been part of a collection called *Deutsche Literatur von Frauen (German literature by women)* which tried to collect as much literature by female authors as possible. As many of these texts are not part of the literary canon, there are no scholarly editions and the creators of the collection had to base their digital texts on first prints or unchanged reprints of first prints. Therefore, the collection is not balanced or representative for the literary production of the period it covers. The collection is copyright free and has been released in TEI-markup on TextGrid-Rep with a Creative Commons-license (CC-BY). The texts of DROC have not been standardized and offer a variability of orthographic norms. The variability is especially marked in nine texts which have been published between 1650 and 1800. If ananalysis of only standardized texts is required, then these can be filtered out via the metadata of the corpus.

4.2.2. Creation of the Corpus

The corpus DROC comprises 90 fragments of different novels. The novels were randomly selected from 450 available novels of the TextGrid repository. The sentence detection component of the Apache OpenNLP ([Baldridge, 2005]), trained on the TIGER corpus ([Brants et al., 2002]),was applied to annotate sentence boundaries in the selected novels. Then, for each novel, a randomly sampled sentence index was selected and the fragment was extended in both directions until the beginning of a chapter and the end of a chapter was reached. In some occasions, where no structural information about chapters was

[3]this section is largely based on [Krug et al., 2018a]
[4]`https://textgridrep.org/`, accessed 20.05.2020

available, our annotators manually selected sentences that indicate the beginning of a coherent passage in the novel and therefore simulate an artificial border. The resulting fragments had an average length of 201 sentences. This procedure was implemented to assure that for all pronominal references either the proper nouns or the common nouns were part of the selected sentences. The annotation process can be depicted as follows: The document were preprocessed with a rule-based script, developed with UIMA RUTA ([Kluegl et al., 2016]), in order to generate suggestions that both of our annotators could later either accept or change. Therefore, the corpus was created semi-automatically with initial support. The novels were annotated in ATHEN, a selfmade desktop application based on the eclipse RCP4 framework, further described in chapter 5 [Krug et al., 2018b]. After the annotators finished their pass over the documents, resulting inconsistencies were resolved together in order to get a clean version of the annotations.

4.2.3. Annotation Guidelines

This chapter presents the annotation guidelines in a three step process. At first, it is described which references were annotated, followed by the description of the resulting phenomena that occured, when dealing with coreference resolution and some borderline cases in DROC. This section is concluded with the guidelines for the annotation of direct speech utterances along with their speakers and addressees.

4.2.4. Annotated Character References

The annotation of character references follows a single rule:

Mark every text snippet in the novel that references a (literary) character.

Furthermore, it was decided not to mark the complete nominal phrase surrounding the reference, but to only mark the heads. Following this rule, the resulting phrases can be classified into the following subcategories:

Proper Noun Proper nouns, for example forenames, surnames or family names. These names can also refer to entities that are not part of the fictional world (e.g. another author, historic persons, etc.) In our schema, the text snippets representing proper nouns are marked as "Core". Sometimes a "Core" snippet is only a part of a reference (As shown in figure 2, where "von Padden" is the Core snippet of "Ritterschaftsrätin von Padden").

Heads of common noun phrases - Appellatives A head of a common noun phrase can be an arbitrary composite consisting of:

- Occupational titles (e.g. "Bäcker" – "baker")
- Relational expressions (e.g. "Mutter" – "mother")
- Gender terms (e.g. "Mann" – "man")

- Different titles (e.g."Graf" – "earl")
- Action terms (e.g. "Spaziergänger" – "stroller")
- Defamations (e.g. "Idiot" – "idiot")
- Substantival verbs (e.g. "Rufende" – "shouter")

This listing is not complete, showing the complexity of this class. Annotations of this kind were marked as "AppTdfW" (Appellativ, Teil der fiktionalen Welt) if they are part of the fictional world or as "AppA"(Appellativ, Abstraktum) if they refer to generic or abstract entities that are not part of the fictional world.

Pronouns This category, marked as "Pron", comprises all sorts of pronouns, the most prominent examples being:

- Personal pronouns (e.g. "er", "sie," – "he", "she")
- Possessive pronouns (e.g. "seine", "ihre" – "his", "her")
- Reflexive pronouns (e.g. "sich" – "himself", "herself", "themselves")
- Relative pronouns (e.g. "der", "die" – "who")

For each resulting character reference, the following features were marked:

- Type: one of "Core", "Pron", "AppTdfW" or "AppA", as described above
- Range: (used only for Cores) span of character offsets for the identification of the core text snippet. (This can be seen as an adaption of the inner and outer span labeling schema that was applied at GermEval 2014)
- Number: singular or plural
- ID: a unique identifier for each entity appearing in the text, used to represent coreference.
- Pseudo: This means that the person is mentioned in the text, but does not take part in the action or does not exist. An example for this case is given:

  "War nicht auch Cromwell erst in hohem Alter nach vergeudeter Jugend erweckt worden zum Dienste Gottes?" [a].

  [a]"Only in his old age and after wasting his youth Cromwell was called to serve God, wasn't he?" [Bleibtreu, 1888]

  Both, Cromwell and Gott, are identified as pseudos, because both are not taking part in this novel's action.
- Uncertain: A boolean flag that can be set by the annotator if the decision is unclear.

4.2.5. Annotated Coreferences

With the definition of the character references, the annotators had the task to assign a unique identifier to each entity in the text, and to reuse this Id for each mention of an entity. To enable an easier comparison of DROC to existing corpora with annotated coreference information, we discuss a selected list of coreferential linguistic phenomena and elaborate whether we marked them as coreferent or not.

Coordination and plural references Plural references are included if the phrase that is required to mark them does not consist of multiple smaller references. Therefore our annotations are not hierarchical.

Split Antecedents Split antecedents, that is plural references which can only be mapped to more than one reference, are not marked. (e.g. "sie" in "Effi und Innstetten planten eine Reise, sie...")[5]

Expletives Expletives, such as "It" in "It is raining" are not included in DROC.

Appositions and Predicatives Appositional references (e.g. "Otto, ihr ältester Sohn,..."[6]) as well as references in predicative position (e.g. "Er ist Bäcker"[7] are (usually) marked as coreferent.)

Bridging Anaphora Bridging anaphora, such as the relation between tire and bicycle in "I bought a bicycle. A tire was already flat", are not marked within DROC.

Discourse The information whether an entity is discourse new has to be parsed from the ID feature of the references.

We conclude this section with a prototypical example taken from DROC. In this example all references are marked by brackets, numbers in superscript denote the entity id and all annotations in subscript denote the type of the reference:

$[Bekannte]^1_{AppTdfW}$ traten zu $[ihnen]^2_{Pron}$ heran und das Gespräch war unterbrochen. $[Michael]^3_{Core}$ fuhr mit $[Kaethe]^4_{Core}$ in einer offenen Droschke, in der milden Märznacht, nach Hause. $[Ihre]^4_{Pron}$ Blicke hingen am gestirnten Himmel, die $[seinen]^3_{Pron}$ an $[ihrem]^4_{Pron}$ Antlitz. In $[Beiden]^2_{Pron}$ klang die Stimmung von $[Tristan]^5_{Core,pseudo}$ und $[Isolde]^5_{Core,pseudo}$ nach.[a]

[a]"Friends came up to them and made their conversation stop. Michael went home with Käthe in an open hansom through the mild March night. Her eyes focussed on the starry sky, his

[5]Effi and Innstetten planned a trip, they...
[6]Otto, her oldest son
[7]He is a baker

eyes focussed on her countenance. Both of them reveled in the mood of Tristan and Isolde." [Dohm, 1894]

4.2.6. Annotated Direct Speech

Direct speech passages and every text section enclosed by single or double quotation marks are annotated. Such annotations range from opening quotation marks till closing ones – both included. In most cases these are french quotation marks (»«), infrequently dashes. To every annotation one (or in rare cases more) speaker and addressed character references is/are assigned. If it was not possible to determine who speaks or who is addressed, they are marked as "unknown". The annotation process obeys strict rules. If speaker and addressed reference are connected to the relevant direct speech with a communication verb, then these entities are labelled. If not, we marked direct addressees within the direct speech which are not pronouns (e.g. "..., my dear friend"). If something like that does not exist either, the last mention of speaker and/or addressed person which lies outside of direct speeches was annotated, independent of it being a noun or pronoun. This has implications for experiments that use this dataset for automatization (see chapter 7. The evaluations should always take the entity id into account and only use the character reference as identifier if clear rules are known (an example for this is that the addressee which is located inside a direct speech can be evaluated on a per reference base, while the speaker in general should be evaluated against the entity id). Apart from real direct speeches every text section within quotation marks is annotated. This might be names of places, quotations or thoughts. (...unten in der »Gelben Straße«...[8])[Dauthendey, 1923]. In this case, the category, which is set to "directspeech" by default, was changed. The following categories are defined:

- **Thought** (113 instances)
- **Citation** (e.g. quotations of absent characters or of other fictional works, 144 instances)
- **Fictionalspeech** (speeches of a text entity that is not labeled as a reference, e.g. "my heart says...","roses say...", 19 instances)
- **Name** (e.g. place names, 88 instances)
- **Song** (2 instances)
- **Other** (if further classification is not possible, e.g. a word highlighted with quotation marks by the author for accentuation, 99 instances)

Sometimes direct speeches are not marked up by quotation marks. In rare cases, direct speeches are labeled by dashes or even without any marker. These cases have been annotated either way.

[8]...down in »Yellow Street«...

4.2.7. Inter-Annotator Agreement

There are multiple ways to measure an inter-annotator agreement (IAA). For 12 documents that were labelled by both annotators and measured the IAA for character reference annotation and for coreference resolution. For evaluating the quality of the character reference annotation we only took the annotated span into account and calculated Cohens Kappa [Cohen, 1960]. This was done on a per token basis and the output of each annotator was conveted into a sequence of B-I-O labels. A measurement on a per token basis awards the annotator for not marking a token as a character reference on top of the rewards for marking the same span. This yields 31.185 instances and resulted in a Kappa κ of 94.3%
On the same documents, the IAA of the assigned coreference clustering was measured using the MUC-6 and B-Cube scores [Luo, 2005]. The evaluation resulted in a MUC-6 F1 of 88.5% and a B-Cube F1 of 69%. In order to get an idea about the pure quality of the agreement on the task of coreference resolution, we added dummy-references so that both annotators had the same amount of mentions and treated them as singletons. The B-Cube metric punishes singleton clusters which explains the much lower score compared to the MUC evaluation. Removing unmatchable annotations yields a MUC-6 F1 of 92.4% and a B-Cube F1 of 76%. For the final version of DROC, the documents were annotated by one annotator and afterwards both annotators revised the documents together to guarantee a corpus of high quality. The code for the evaluation, as well as the documents that were used for the measurements can be downloaded from DROCs git repository[9].

4.2.8. Corpus Statistics

DROC contains 90 fragments of different novels. The corpus comprises about 393.000 tokens as determined by the tokenizer script of the TreeTagger [Schmid, 1995]. On average each fragment consists of 4.368±2.334 tokens and 202±131 sentences. We manually annotated 52.079 character references with the majority of 65% being pronouns (34.060). About 23% (12.005) of the references belong to the type ”appellative” and the remaining 12% (6.013) are ”Core” references. These 52.081 references are clustered into 5.288 entities, therefore on average 10 references per entity and 59±31 entities per document. Compared to the statistics from the study in [Kabadjov, 2007], pronouns in DROC appear more frequently, with a proportion of 65% compared to 44% evaluated by Kabadjov, with the amount of proper nouns being almost constant with a small increase from 10% to 12% in DROC. With 35 of those 90 fragments written by female authors and the remaining 55 written by male authors a slightly imbalanced 40%-60% gender ratio is present. As shown by table 4.1, most novels of DROC were published between 1801 and 1900.

[9] `https://gitlab2.informatik.uni-wuerzburg.de/kallimachos/DROC-Release`

Table 4.1.: Overview of the amount of novels published during an epoch of 50 years. It can be seen that most novels were published during the 19th century.

Epoch	1651 - 1700	1701 - 1750	1751 - 1800	1801 - 1850	1851 - 1900	1901 - 1950	1951- 2000
Amount novels	2	3	4	31	35	14	1

4.3. Kindlers Expert Summaries

A secondary and very important resource for this thesis are summaries of novels that are taken from the online version of the Kindler lexicon [10]. This lexicon offers, among other information (such as background information about authors), summaries for many novels. We were able to gather a corpus comprising 213 summaries of our 450 novels present in the the *Kernkorpus*. The summaries itself show a high variance in their length. The shortest summaries only comprise a single sentence while the summary of *Don Quijote* spans about 2500 tokens.

All 213 texts have been manually annotated with character references, coreferences, as well as relations between (a pair) of character references. The annotation of the character references and coreferences was executed using the guidelines presented in section 4.2.4. This section continues by describing the guidelines for the annotation of the relations and concludes with statistics about the resulting dataset.

4.3.1. Guidelines for the Annotation of Relations Between characters

A relation can be considered as a link from one character reference (which is modelled as *Agens*) to another reference (which is called *Agens2* in the data structure). This does already state, that a relation is considered to be a **directed** link. The annotator is instructed to only add a relation between two character references if the text does *explicitly* mention this relation and then assign the Agens and Agens2 in the manner which is described in the text. For example if the text says: *"Peters Mutter geht einkaufen"*[11], then the annotator is expected to create a relation that begins at *Peter*, ends at *Mutter*. The Agens of this relation is *Peter* and the Agens2 is *Mutter*.

Naturally, relations can be typed using a label. In the previous case, a fitting label would be of the kind *hasMother*. We followed a categorization introduced by the work of [Massey et al., 2015] and annotated based on four broad categories for the relations, **family relation, love relation, professional relation and social relation**. Those four categories are yet again split into more detailed sub categories. In total, the labelset comprises 57 distinct relation types that are order hierarchically. The subtree for the family relations is shown in figure 4.1, the subtree for the love relations in figure 4.2, the tree for professional relations in 4.3 and the tree for social relations in 4.4

[10]`www.kll-online.de`, accessed 20.05.2020
[11]Peter's mother goes shopping

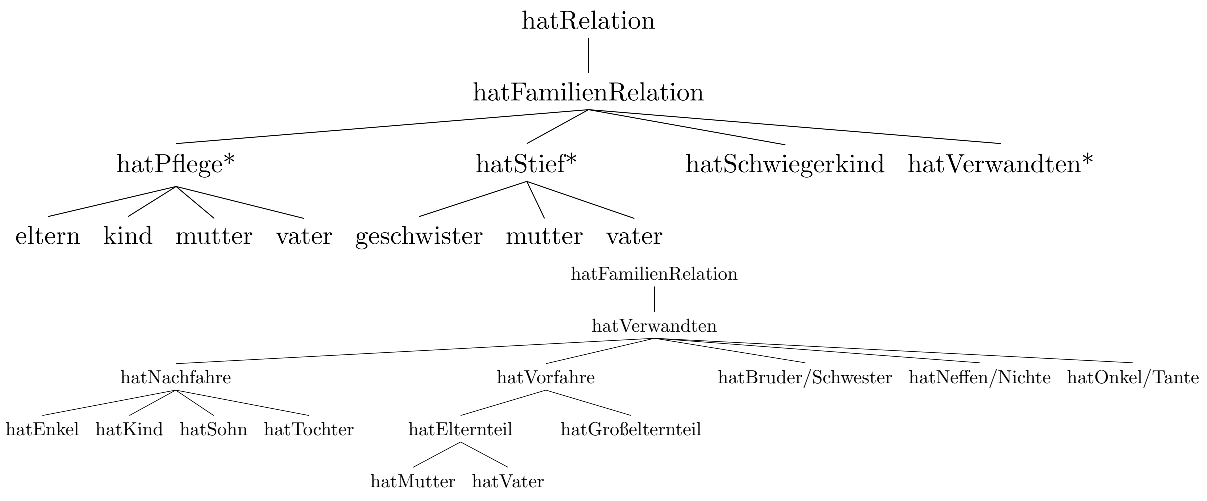


Figure 4.1.: The sub tree for family relations

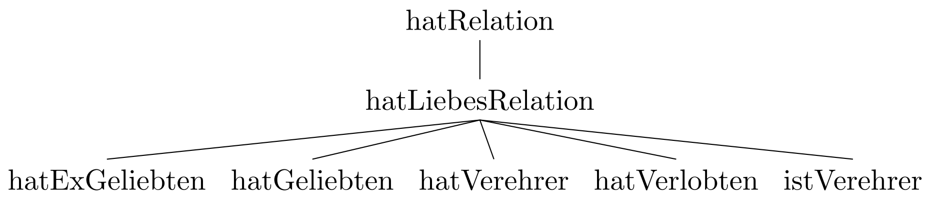


Figure 4.2.: The sub tree for love relations

4.3.2. Statistics of the Kindler Data set

This section shows the statistics that originate from the Kindler data set. It comprises about 85.000 tokens with 3.019 sentences. The documents vary greatly with respect to their size (401±258 tokens). The distribution of character references is different compared to the novels, with about 40% pronouns, 40% appellatives and 20% core references. The total amount of annotated references is about 13.000 that are grouped into 2.826 entities. The corpus contains 2.270 relations of which about 42% are family relations, 35% are social relations, 17% are love relations and 6% are professional relations.

4.4. Active Learning to Annotate Relations in German Novels

Annotating relations in the text of the novel is cumbersome as there are sometimes entire pages without an explicit marker of a relation. However, since a large portion of relations are stated within a sentence, we decided to experiment with Active Learning in order to sample sentences in which relations are to be expected. Since the philosophy of ATHEN did not support Active Learning of any sort, an additional dedicated tool was created that was suited to annotate with a human that is integrated in an Active Learning loop. The tool is depicted in figure 4.5

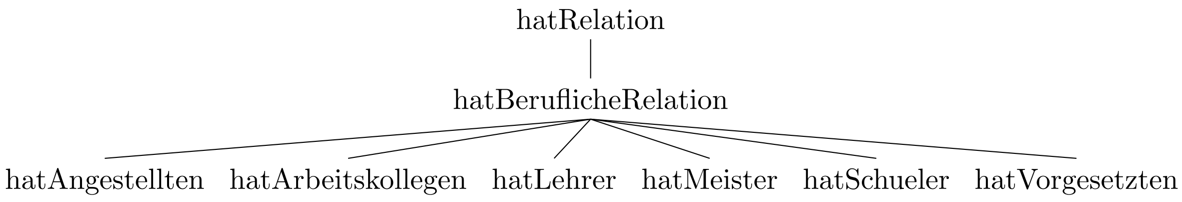


Figure 4.3.: The sub tree for professional relations

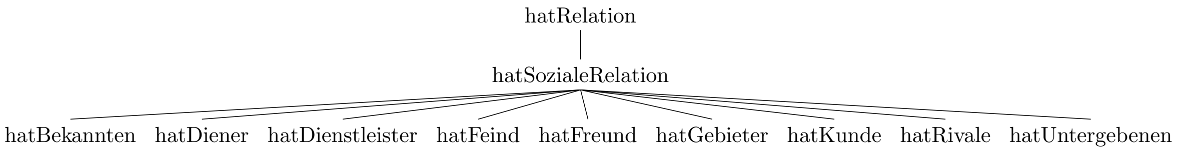


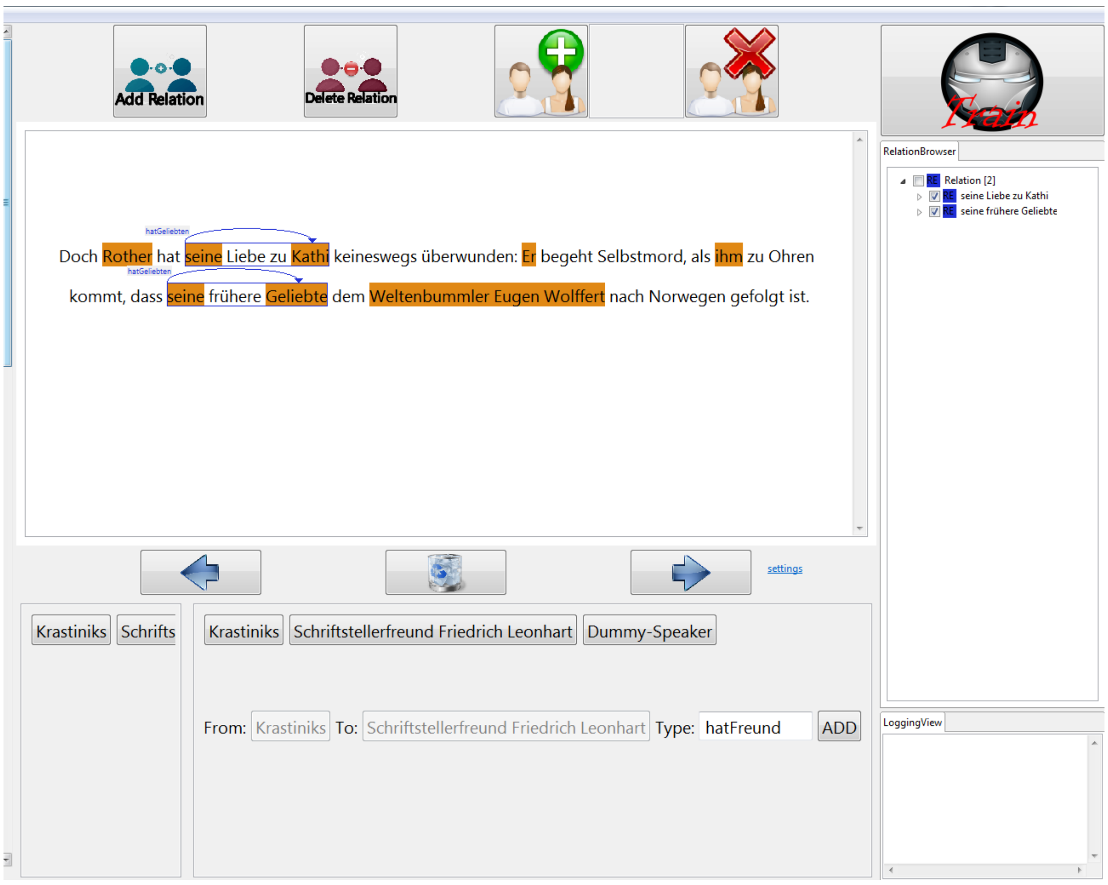
Figure 4.4.: The sub tree for social relations

A small subset containing about 20 seed instance were labelled by hand and a MaxEnt classifier was used to propose new data based on the Query by Uncertainty schema. An overview to Active Learning is given in [Finn and Kushmerick, 2003]. The environment gets as input sentences from 312 different novels from the TextGrid digital library, Project Gutenberg, and 215 summaries. The user can then proceed to mark text and click on *Add Relation* in order to add a new relation or delete an existing relation by pressing the according button. The other two buttons in the top of the tool are to add and delete character references that have been marked incorrectly by the preprocessing component. The top right button is used in order to manually start the new training process. Once this is done, a background thread is then used to label all the sentences with the new model. All sentences that have been labelled that way are sorted by the confidence of the classifier. The user can navigate between those snippets using the arrow buttons, or discard a snippet using the symbol with the trash can. If a relation in the text is selected, the assignment of both agents is done in the same manner as in ATHEN, the tool lists all candidates for both slots and the user can select the appropriate candidate.

4.5. Inter Annotator Agreement for the Annotation of Relations

Two datasets with manually annotated relations became available in this manner, the first data set was created by the usage of the Active Learning environment, while the second dataset was created by labelling the 213 Kindler summaries using ATHEN. The datasets are from now on referred to as Summaries I and Summaries II. The inter-annotator-agreement (IAA) between Summaries I and Summaries II was measured in two ways:

1. A true positive appears when both annotators mark the correct span of the annotation as well as the correct label and the correct arc direction where a correct



Doch Rother hat seine Liebe zu Kathi keineswegs überwunden: Er begeht Selbstmord, als ihm zu Ohren kommt, dass seine frühere Geliebte dem Weltenbummler Eugen Wolffert nach Norwegen gefolgt ist.

From: Krastiniks To: Schriftstellerfreund Friedrich Leonhart Type: hatFreund

Figure 4.5.: The user interface of the tool that was used to label text snippets that were selected using Active Learning for the Annotation of relations

arc links the two entities in the direction as it is expressed in the text (labelled inter-annotator agreement).

2. A true positive appears when both annotators mark the correct span and arc direction of the relation (unlabelled inter-annotator agreement).

The results of the IAA are depicted in table 4.2

Additionally, we determined 55.5% as the normalized Cohen's Kappa between our annotators. The results for the IAA are surprisingly low. At the current point in time, we suspect, that by labeling only short snippets of the text, the annotator can not interpret relationships which require additional knowledge that is obtained by reading the text. The complete summaries may also be more difficult because the annotator needs to read the text completely and might use background knowledge to annotate relations which are only implicit in the text.

Table 4.2.: The results for the inter annotator agreement between two scenarios to label relations. The first used short text snippets that were selected using Active Learning and the second scenario labelled entire texts.

Evaluation Method	Precision	Recall	F1
Unlabelled IAA	75.6%	43.7%	55.4%
Labelled IAA	60.9%	35.2%	44.6%

4.6. Annotation of Interactions in Parts of DROC

For 40 documents of DROC, we labelled interactions between the previously annotated character references. Each annotation comprised two character references and a label. Depending on the label, the role of the references changes. This serves as an additional dataset for relation detection (for the experiments, see chapter 9), but compared to the family and social relations that were labelled in the summaries or in snippets of novels, their impact differs. While family relations can mainly be utilized to label edges between entities in character networks, interactions between entities can be reflected in the edges of a character network. While in many novels, interactions can be modelled using direct speech and dialogs between entities, this set of interactions is not complete. Adding interactions extracted from verb frames should sensitize a character network and enable a more detailed depiction of relations between entities.

The annotation of interactions (within a sentence) is dominated by the finite verb that is present, however it is not the only criterion. In this sense, two labels for interactions are introduced:

- Interact: It should contain all actions that both entities perceive, regardless of whether they are in direct or indirect speech or simply described.
- Observes: It should contain all actions of which the ”observed” entity is unaware, no matter if it is in direct or indirect speech or simply described.

Since not every mentioned interaction can be considered equally reliable, the labelset introduces derivated labels, examples for them are given in table 4.3:

- **Neg:** - The interaction is expressed with a negation.
- **Future:** - The interactions takes place in the future.
- **Subjunctive:** - The interaction is expressed in subjunctive.

An instance of an interaction is not limited to a single derivated label, but instead can have more than one. The sentence:

Ich werde ihn nicht beobachten.[a]

[a]I will not observe him.

Table 4.3.: The different derivations for an interaction.

Derivation	**Example (German)**	**Example (English)**
future	Ich werde sie treffen	I will meet her
subjunctive	Wenn ich ihn treffe	When I meet him
neg	Sie sah ihn nicht an	She did not look at him

will be labelled as an observation both, in future and negated.
Usually the verb can be considered to be the deciding factor, it is not sufficient to just use the verb and its arguments:

Er überraschte **sie** mit Blumen. (interaction)[a]
Er überraschte **sie** durch seine Tapferkeit. (observation)[b]

Er erkundigte sich bei **ihr**. (interaction) [c]
Er erkundigte sich nach **ihr** (observation) [d]

[a]He surprised her with flowers.
[b]He surprised her with his bravery.
[c]He inquired with her.
[d]He inquired about her.

In this case, it depends on the meaning of the verb which can be derived by considering the prepositional object and the according preposition.

For some verbs however, the allocation is rather unambiguous, consider the following examples for the label *interaction*:

- **Verbs of communication:**
  Er erkundigte sich bei **ihr**. (interaction)
  ... weil **sie** von allen Seiten **ihr** seinen Wert pries. (interaction)
- All visible actions, that can be recognized by the other entity[a]
  Sie sahen sich an (interaction)[b]
  Die Mutter, die ihm auch ihre zweite Tochter vorstellte (interaction)[c]
- If the reference is "beide" (both), then usually an interaction can be marked (in this case, both agents are depicted by the same reference):
  Beide hatten mit der Situation zu kämpfen.[d] (interaction)
- If a prepositional object which starts with "mit" ("with") is available, and this prepositional object contains a reference, then it is likely that an interaction can be found in this construction:
  Ich war dort mit dem Kurfürsten.[e] (interaction)

[a]e.g. sich verneigen, auf jmdn. zueilen, jmdn. heiraten, sich um jmdn. bewerben, jmdm. die Hand geben, jmdn. tragen, nach jmdm. schicken

[b]They looked at each other.
[c]The mother, who introduced him to her second daughter.
[d]Both had to struggle with the situation
[e]I was there with the Elector.

Analogously, for the label *observation*:

- Verbs, which describe thoughts and feelings:[a]
- Verbs, which describe thoughts and feelings:
  Er liebt sie.[b] (observation)
  (but: Sie liebten sich.[c] (interaction))
  Des Gastes Heiterkeit theilte sich den anderen mit.[d] (observation)
  Der Ritter errieth das Wort, das auf den Lippen der Edelfrau erstarb.[e] (observation)
- Verbs that are associated with sensory impressions, if they are one-sided[f]:
  Er bemerkt sie.[g] (observation)
  (but: Sie sahen sich an.[h] (interact))
- Actions that are not visible to an external entity[i]

[a]z.B. jmdn. lieben, jmdn. ehren, jmdn. kennen, jmdm. den Vorzug einräumen, etwas (gedanklich) annehmen, jmdm. etwas sein (from: Subjekt, to: jmdm., observes), jmdn. anerkennen, jmdm. ein willkommener Gast sein, jmdn. verleugnen
[b]He loves her.
[c]They loved each other
[d]The guest's serenity shared with the others
[e]The knight guessed the word that died on the lips of the noble woman
[f]e.g. sehen, erblicken, anschauen, bemerken
[g]He notices her.
[h]They looked at each other.
[i]e.g. jmdm. etwas verbergen, jmdm. etwas überlassen, jmdm. etwas widmen, sich jmdm. unterwerfen, jmdm. etwas schulden, nach jmdm. schicken, jmdm. etwas erlauben, jmdm. ein (gedankliches) Rätsel lösen, jmdm. etwas entziehen; jmdm. Gelegenheit geben, jmdm. (z.B. die Stütze) rauben

It is noteworthy, that interactions can not only be marked between personal pronouns, but also between possessive constructions or genitive constructions, even though the head of the phrase is not necessarily a character reference:

Er bemerkte **ihre** schönen Hände (observation)[a]
Er bemerkte **Lydias** schöne Hände (observation)[b]

[a]He noticed her beautiful hands
[b]He noticed Lydia's beautiful hands.

4.6.1. Statistics about the Dataset

In total, the corpus contains 9299 sentences in which 6662 interactions are annotated. Of these, 3297 are marked as observations and 3365 as interactions. Most of these annotations carry the pure label without any derivations, but the most frequent derivation is the subjunction. The distribution is shown in figure 4.6. With about 0.6 annotations per sentence, the yield is comparable to that of manually annotating family and social relations.

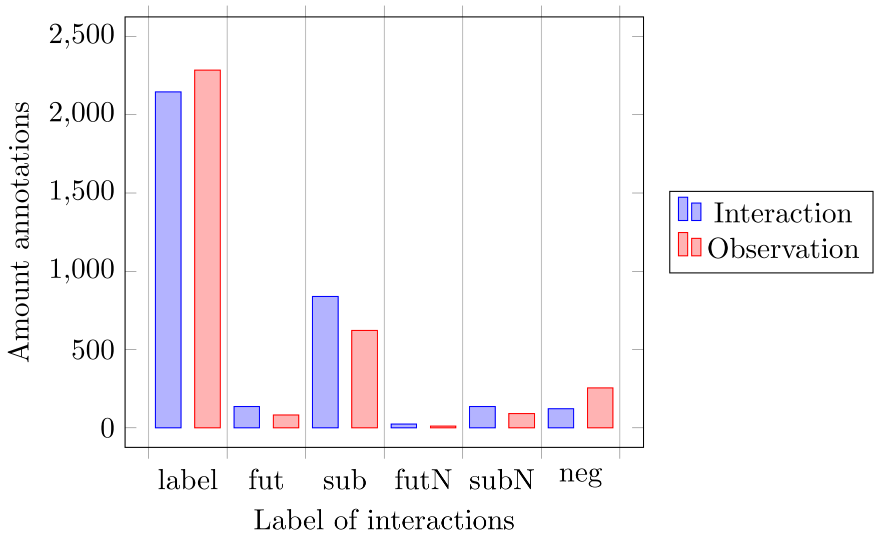


Figure 4.6.: Number of manually annotated interactions and their derivations

4.7. Gazetteers and Further Lexical Resources

This section describes the resources that were used during this thesis. The resources are grouped into different categories. The first category is of the category *dictionary*, it contains the available dictionaries for German that were utilized. The second category contains the semantic resources, that found its use in any of the developed algorithms. This comprises mostly gazetteers and their origin. The third category are resoures that were generated during this thesis, so they are semi automatically created and contain mistakes.

4.7.1. Dictionaries

For this thesis, a single dictionary was primarily utilized (the lexicon of the RFTagger), nevertheless, this section goes over the different German dictionaries since they are available as well.

Lexicon of RFTagger: This lexicon is bundled with the model of the RFTagger, it was generated by analyzing the data (the tokens and their labels) of the TIGER corpus. These tokens were automatically inflected with morphological information using a finite state machine (however it is unreported which one, good candidates might be SFST [Schmid, 2005] or SMOR [Schmid et al., 2004]). The bundled lexicon contains about 3.400.000 different morphological forms alongside their case, their number, their gender and their POS-Tag.

Morphy: The lexicon[12] which comes with the software *Morphy* [Lezius, 2000] contains about 6.000.000 morphological forms of German words, each entry having the same amount of information as the entries of the lexicon of the RFTagger.

The German Duden: A version of the Duden[13], which comprises about 210.000 entries of German lemmata was crawled during September and October of 2015. Each lemma itself is represented by a multitude of different morphological representations.

The dictionary of the jwcdg: The constraint based dependency parser that was developed by the university of Hamburg [Gerdes et al., 2013] features a lexicon similar to the one of the RFTagger, however name lists are provided with more details (locations, person names etc...).

Wiktionary: The Wiktionary[14], similar to the German Duden contains entries for German lemmata and morphological inflected forms. The dump as of August of 2019 contains about 550.000 pages for German words.

By the usage of dictionaries and morphological analyzers it is possible to split compound words, which is a necessity for the detection of character references (more precisely for the detection of compound appellatives).

4.7.2. Gazetteers and Semantic Resources

This section goes over the gazetteers that were used for the rule-based character reference detection, the category determination, the extraction of family relations as well as the coreference resolution, when Metadata of entities was analyzed. During this work, especially GermaNet [Hamp and Feldweg, 1997] stood out and was very helpful, especially when categories of nouns or verbs were required (as is the case for character reference detection or for the creation of a list of communication verbs). Additional resources, such as lists of first names (female names and male names), lists of titles and nobility

[12]`https://github.com/languagetool-org/german-pos-dict`, accessed 20.05.2020
[13]`https://www.duden.de/`, accessed 20.05.2020
[14]`https://de.wiktionary.org/wiki/Wiktionary`, accessed 20.05.2020

predicates, were manually created using Wikipedia[15]. Additional good resources for the acquisition of lists was *taschenhirn*[16] or miroso[17].

4.7.3. Semi automatically created resources

The extension of the gazetteers that are presented in section 4.7.2 was done in the context of character reference detection and is shown in section 6.4.1. Other automatically created resources that were used during this work are word embeddings created from about 1800 novels extracted from project Gutenberg using the Word2Vec [Mikolov et al., 2013] and the GloVe [Pennington et al., 2014] algorithms.

4.8. Selection of appropriate Preprocessing Components for German Literary Documents

This section serves the purpose of evaluating existing tools and giving an estimation about their quality as well as their problems that arise when they are applied to literary text. Since there are no data sets available for German historic novels, the comparison of different tools has to be conducted on small, self labelled data sets. The results of this section serve the justification of the initial configuration of the Kallimachos Pipeline configuration (for more information, as described in section 2.2. Some of the experiments were conducted as studies during students Bachelor as well as Masters thesis. The according sections will explicitly cite these.

4.8.1. Tokenizing and Sentence Splitting

The literary domain is (at least compared to other domains such as medical discharge letters) a rather forgiving domain in terms of complexity of tokenization and sentence splitting. However since it is the first step of the pipeline, mistakes in those engines can usually not be captured later on and should therefore be avoided.

During the majority of the experiments, the tokenizer and sentence splitter of the OpenNLP framework was used. It is easy to use and very fast. Internally, it makes use of a Maximum Entropy classifier, which predicts for every character (only characters that are in tokens as determined by a previously applied whitespace tokenizer are considered), whether at a given character index a token needs to be split or not. The sentence splitter repeats this decision but with (special) tokens instead of characters, and decides for sentence boundaries. The components (using the available German models) however have some issues, when applied to literary text (however some might not matter):

- It deals poorly with apostrophes (e.g. ”er's”, which is short for ”er es”), not creating additional tokens.

[15]e.g. the list of noble titles: `https://de.wikipedia.org/wiki/Adelstitel`, accessed 20.05.2020
[16]`https://www.taschenhirn.de/`, accessed 20.05.2020
[17]`https://www.miroso.de/`, accessed 20.05.2020

Table 4.4.: The distribution of manually labelled phrases from snippets of 22 novels. This table is taken from the work of Tritscher.

Syntactic Function	**Subject**	**Accusative**	**Dative**	**Prepositional Phrase**
Amount	790	153	488	615

- It has only a very basic capability to detect abbreviations. While not being very frequent in novels, the tokenizer fails to detect the abbreviation in ”Mrs. Bumble” and instead creates a separate dot which in turn ends a sentence.
- The sentence splitter never ends a sentence on semi colons or colons.

4.8.2. Part of Speech Tagging and Morphology

The TreeTagger and the RFTagger are both very fast and very reliable. Even though no detailed evaluations have been made, it is safe to assume, that their quality exceeds well over 90% tagging accuracy. Speaking of the remaining errors, in the case of POS tagging, both taggers do sometimes confuse articles with pronouns (which is sometimes a very hard task) and have trouble to differ common nouns (tag: NN) with proper nouns (tag: NE).

The quality of the morphology (especially gender and number of a token) for unknown words is much lower than its POS-tagging quality. For example if a token is just a last name (e.g. ”Innstetten”), then there is usually no clear evidence in the same sentence, whether it is male or female in this scenario, but instead requires more context. Both taggers act solely on the sentence level and therefore can not make use of these constraints. This is especially hindering for the coreference, because it especially relies on morphology for the resolution of pronouns to other noun phrases. It is also noteworthy, that the RFTagger always predicts the syntactic gender and number of a token (e.g. for the token ”seine” it is singular and female), while for the coreference you need a semantic morphology (singular, male). During the course of this work, a separate list with the semantic morphological form was created manually.

4.8.3. Dependency and Constituency Parsing

The analysis of both qualities has been done as parts of the Bachelor thesis of Julian Tritscher [Tritscher, 2016].

Starting with the analysis of the Dependency Parsers. Tritscher analyzed three parser and their results on a text sample of about 10.000 tokens, taken from 22 different German novels. He manually labelled subjects, dativ, accusativ and prepositional objects as phrases and evaluated based on the labelled entity score as shown in section 3.1.2. The distribution of his labelled phrases is shown in table 4.4

He considered three different Dependency Parser:

- Mate Parser [Bohnet, 2010]: A transition based parser which predicts POS tags

Table 4.5.: The prediction score (only the micro labelled entity-F1 is reported in %) of the three dependency parsers on the manually labelled dataset. This table is taken from the work of Tritscher.

Syntactic Function	Subject	Accusative	Dative	Prepositional Phrase
Mate	69.8	39.1	58.0	67.7
Malt	72.6	45.4	58.3	63.0
ParZu	**76.2**	**79.3**	**66.2**	**70.6**

and dependency relations at the same time. It comes with a pretrained model for German

- Malt Parser [Nivre, 2003]: Similar to the Mate parser, this parser is based on a transition system as well. It does not come with a German model, this is why Tritscher trained a model based on the TüBa-D/Z treebank [Telljohann et al., 2006].

- ParZu [Sennrich et al., 2009]: ParZu is a rule-based Dependency Parser, which can be seen as a German adaption of the Pro3Gres [Schneider, 2008] system based on the German Dependency Grammar of Foth [Foth, 2006].

The results of Tritscher are shown in table 4.5 [18]. In his evaluations, ParZu outperformed the other parsers by a large margin on all four considered categories. These results match the ones that are presented by the authors of ParZu. In their work, they also include the MSTParser [McDonald et al., 2006], which is a graph based dependency parser. Tritscher also compared the results to the Berkeley and Stanford constituency parsers [Chen and Manning, 2014] and found that both constituency parsers were greatly outperformed by all three dependency parsers. This is not necessarily bad news, since both constituency parsers also are significantly slower than the dependency parsers (see section 2.2.1).

One big disadvantage of the Mate Parser, is that it tends to predict more than one subject per finite verb (this often results in "sie" (she) being misclassified as a subject instead of as an object in accusative position) and has trouble to differ between accusative and dative objects. The availability of a good dependency tree is of great importance for a correct detection of speaker as well as for the coreference resolution and relation detection.

[18]Note: It is not straightforward to evaluate a dependency parser againt noun phrases, since the parser predicts edges in the first place. However these can be converted with a high reliability using the algorithm presented in [Kübler et al., 2009]

4.9. Development of unpublished Preprocessing Components for German literary texts

4.9.1. Tokenizer and Sentence Splitting Component

In order to resolve the issues that arise with the OpenNLP components (see section 4.8.1) a rule-based system for the detection of tokens and sentences was developed. The tokenizer assigns a label to every character in a text (B, I and O) and concludes by building the tokens from these labels. The tokenizer consists of three passes through the characters of the text:

Initial Pass: In this pass, all whitespace characters are assigned an **O** label, all digits or textual characters either a **B** or *I* token and all ambiguous characters are assigned an **UNKNOWN** token. All these checks are performed using regular expressions. Ambiguous tokens are commas, dots, apostrophes and hyphens.

Resolver Pass: For every token that was assigned an **UNKNOWN** token in the first pass, a resolver deals with the ambiguity. A comma can reside in a token if it is surrounded by digits. A dot can either be part of an abbreviation, can be part of a number or can mark the end of a sentence. In order to detect whether the token is part of an abbreviation, the algorithm makes use of a dictionary (see section 4.7) as well as all words that could be detected by the Initial tokenizer pass. Using these tokens, it is possible to apply a number of heuristics which are also prevalent in the Punkt tokenizer system. [Kiss and Strunk, 2006]. A hyphen can be its own token if it is surrounded by whitespace or be part of a token (e.g. ”Hans-Peter” or ”An- und Abreise”). An apostrophe can either be part of a token (e.g. ”gnäd'ge”) or can start its own token (e.g. er's). This disambiguation is done by considering the surrounding characters and tokens.

Final Assignment Pass: In this pass, all remaining **UNKNOWN** tokens are resolved. There may have been tokens that are still **UNKNOWN** just because any neighbouring token was assigned to be unknown as well. After the resolver pass however, these are most likely resolved and an additional pass gets rid of these cases. The remaining unknown tokens are simply assigned to be **B** or **I**, depending on the assignments of the previous character.

These final labels are then used to build the tokens which are returned subsequently. The sentence detection component uses these tokens and creates sentences based of them. A sentence ends on either an exclamation mark, a quotation mark or a dot that was determined to be its own token. There is one special case however. If a sentence contains direct speech, then a sentence might not end on a dot, but on a following quotation mark instead.

4.9.2. Ontology Based Information Extraction

Developed to work with the previous results, this component facilitates otherwise repetitive structures. Its goal is to provide a way to specify and store knowledge in the form of an ontology and by the use of this ontology, additional relations and concepts (in the form of annotations of the type ”Entity”) can be extracted and annotated automatically. For example, the detection of a speaker of a direct speech utterance requires the knowledge, whether there is a verb of communication associated with a character reference. This is usually represented as a concrete path in the dependency tree (in the simplest case it is just a verb → subject edge). The ontology underlying this module can be designed in a way that tasks like this can be defined externally, stored as an ontology and this module extracts relations using this ontology.

Starting with some background of Ontology-Based Information Extraction. OBIE is a subfield of Information Extraction, Daya et al. [Wimalasuriya and Dou, 2010] provide the following definition:

> An ontology-based information extraction system: A system that processes unstructured or semi-structured natural language text through a mechanism guided by ontologies to extract certain types of information and present the output using ontologies.

Alongside the definition, the authors presented a schematic general architecture for an OBIE system (see figure 4.7). An OBIE system gets a textual input alongside a diverse set of lexical resources as its input, a resource which is commonly used is WordNet [Miller, 1995]. The text gets preprocessed and an ontology is created by a human domain expert. The ontology as well as the preprocessed text and the lexical resources serve as input into the information extraction module and the extracted information is stored in a knowledge base or database. This database can then be used to serve queries by users.

Following this overview, the existing systems can be distinguished using five criteria :

1. The information extraction method: Whether the method is rule-based or based on machine learning.
2. Ontology construction: Was the ontology created manually or automatically, is the ontology updated during the extraction process.
3. Text Preprocessing: How much preprocessing is used for the systems
4. Domain: In which domains does the OBIE system excel.
5. Types of extraction: What kind of information is extracted by the system and how is it stored.

The Kylin system [Wu and Weld, 2007] uses machine learning (in particular Conditional Random Fields) which makes use of existing info boxes of articles of Wikipedia to

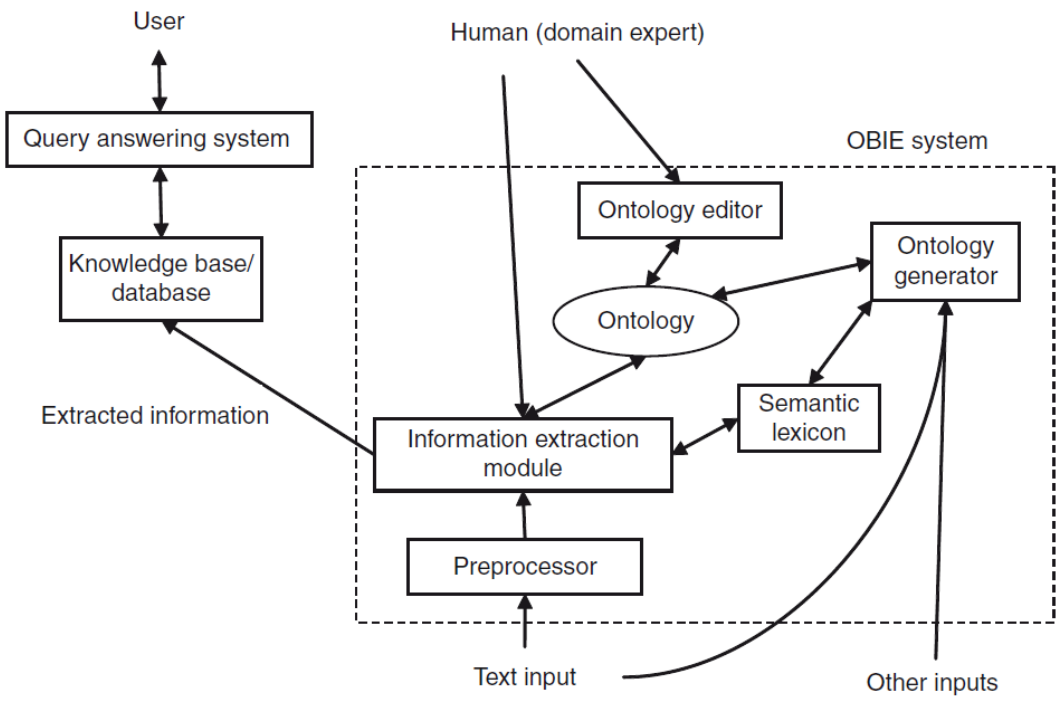


Figure 4.7.: The general architecture of an OBIE system, image taken from [Wimalasuriya and Dou, 2010].

label data and apply these classifiers on Wikipedia articles without info boxes to create new ones.

SOBA [Buitelaar et al., 2006], KIM [Popov et al., 2004] and the system of Saggion [Saggion et al., 2007] extract concepts and relations using a rule-based approach and by the usage of gazetteers. The ontology itself can also be used for slot filling approaches and therefore populate existing ontologies. The information extraction system ONTO-Text [Anantharangachar et al., 2013] uses the StanfordCore toolsuite [Manning et al., 2014] in order to preprocess their incoming text data (web pages from hotels) and populate an ontology describing hotels and their features. In this (reasonably small domain), they achieved an extraction accuracy of about 95% on a small manually labelled dataset. OBIE is still a research topic, in order to facilitate the search of appropriate scientific paper, the work of Almugbel [Almugbel, 2019] propose an OBIE approach to create a semantically annotated corpus of scientific papers. An approach which operates on semi structured data is presented by Rizvi et al. [Rizvi et al., 2018]. In their work, the focus is on the extraction of user relevant information from tables in technical reports. Their approach is based on heuristics and they evaluated their system with an F1 score of about 93%. Another system which is currently in development is the system of Zitnik [Zitnik, 2012]. Using Skip-Chain CRFs [Sutton et al., 2012], a combined and iteratively refined extraction of Named Entities, Relations and Coreferences is proposed which converts a semi automatically enriched ontology into semantic features.[19].

On the other hand, there are systems, which either learn their ontology entirely or at least update them during the extraction process. A prominent example is by the usage of *Hearst patterns* which were used to extract hypo and hypernyms [Hearst, 1992]. Approaches that use clustering algorithms, e.g. the Brown clustering [Brown et al., 1992a] are presented in [Maedche and Staab, 2001]. Generative graphical models with the capabilities of learning concept graphs using Stick-Breaking priors are used in the work of Chambers et al. [Chambers et al., 2010] and a system that learns concept hierarchies using formal concept analysis is presentend in [Cimiano et al., 2005]. Systems, which include an ontology learning mechanism include Text-To-Onto [Maedche and Staab, 2000], DLLearner [Lehmann and Hitzler, 2007] or HASTI [Shamsfard and Barforoush, 2002]

In general, the survey papers of Daya et al. [Wimalasuriya and Dou, 2010], Konys [KONYS, 2015] and Bialek [Białek, 2010] provide a good and broad overview over the existing approaches towards OBIE, while Maedche [Maedche and Staab, 2001] and Drumond [Drumond and Girardi, 2008] present approaches that are centered around ontology learning.

The following section describes the current implementation of the OBIE module and its capabilities. The algorithm is designed to work with German text as input and makes use of syntactical preprocessing until a syntactical parse of the text is available. The algorithm itself identifies concepts and binary relations between concepts and stores them as Apache UIMA annotations. The extraction algorithm is purely rule-based. The ontology does not directly contain extraction rules but the content of the ontology is used to guide the extraction process by providing synonyms and information about frames of special verbs. As input, the algorithm requires an ontology, which can be created and edited using WebATHEN[20]. In its core, the ontology consists of concepts and relations. The concepts depict the entities which are of interest to the user and the relations describe the relations between the concepts that are to be extracted. All relations that can be extracted are (at the current point in time) binary relations, that is only two fields can be set by concepts. Since usually a complex expression can not be considered to be binary, more than one chained relation is added to express a relation. Consider the following example:

Sie hatte dunkelbraunes Haar.[a]

[a]She had dark brown hair.

If one wants to extract the relation that describes the hair color of the entity, the following steps have to be done: a) it has to be determined, that the word "Haar" actually belongs to "Sie" and b) that "dunkelbraunes" describes the work "Haar". So in total two binary relations have to be extracted in order to model this expression.

[19]My own impression: This is never going to work. Their approach models the tasks of coreference resolution as a sequence labeling task, which does not even slightly represent the nature of that task, being a clustering.

[20]`http://webathen.informatik.uni-wuerzburg.de/`

The algorithm works in two phases. The first phase matches the concepts to the text and the second phase creates relations between these concepts. Both phases are described in more detail:

Phase 1, Matching of the concepts, to the text: A concept contains meta data that describes the variances that are used to match a concept to the text. The variances can be manually assigned to be strings (these are most likely synonyms) or be described by regular expressions. These regular expressions are matched on the token level of the text. In order to provide flexibility, additional to the current text of a token, the lemma is considered in the match as well. On top, the ontology and the extraction algorithm allow a concept to be converted from different annotation types that are already available in the text. This is the easiest way to connect the OBIE component to results of previous machine learning algorithms. If matches on the token level are not enough, then, by the usage of the prefix ”phrase=” a regular expression against the covering noun phrase can be used in order to identify a concept. Once the concepts were detected, a simple coreference mechanism extends these concepts to relative pronouns. The easiest way to understand this is by the usage of an example:

Der Pullover, der rot war.[a]

[a]The sweater that was red

In order to extract the color of the sweater, an intermediate step which involves a relative pronoun is required. So the coreference mechanism spreads the concepts onto according relative pronouns, which is determined by the usage of the Dependency Parser.

Phase 2, Extraction of Binary Relations between Concepts: After the concepts have been detected, the relation extraction is applied on a sentence level. Each pair of extracted concepts in the first step is considered to be a candidate for a valid relation. For a valid relation, there has to be a user defined relation in the ontology between both concepts or any of their ancestors in terms of the concept hierarchy. If the semantic step was successful, then the syntactic position between the concepts is analyzed. If both concepts are to be found a valid relation by one of three syntactic conditions, then a relation is created:

Predicative Condition: Both concepts are in a subject-predicative position. This condition respects shared subjects and enumerations. It is best illustrated using examples, where the extracted concepts are marked in bold:

Die **Nase** ist **rot**.[a]
Die **Nase** ist **rot** und die **Hand blau**. [b]
Die **Nase**, **Lippen** und **Hände** sind **blass**.[c]

[a]The nose is red
[b]The nose is red and the hand is blue.

[c]The nose, lips and hands are pale.

In the first example, a relation between "red" and "nose" is extracted, this is the base category of this condition. However things can get really complicated if enumerations are involved. In the second example, the verb "is" is shared between two statements and has to be detected as such, while in the third example, the concept "pale" has to be assigned to all three concepts.

Phrase Attachment Condition: Whenever the two concepts are attached to one another by either genitive constructions, prepositional attachments or appositions, then they are considered by this condition. Examples involve:

Der **Mann** mit den gelben Augen reist gerne.[a]
Der **Bommel** des **Schals** ist flauschig.[b]
Er trägt einen **Schal**, einen **bunten**.[c]

[a]The man with the yellow eyes likes to travel
[b]The bobble of the scarf is fluffy
[c]He's wearing a scarf, a colorful one.

In all examples, only the primary relation is marked. In order to extract the full relation, an additional binary relation has to be added. In the first example, a PP attachment depicts the relation between a man and his eyes. In the second example, the bobble and the scarf are in a genitive construction and in the third example, an apposition connects "colorful" with "scarf"

Dependency Paths Condition: This is the most complex (and also most error prone) condition for the extraction. It creates all dependency paths between all tokens of the concepts, and checks whether at least one dependency path fullfills one of 16 rules. In some occasions, this includes the same conditions as previous conditions, but extracting the same relations by using different aspects renders the algorithm more robust to preprocessing errors in either the chunker or the parser. The rules for this are attached in appendix D.2.

After these relations have been extracted, the second stage of the relation extraction begins. This time, relations are to be extracted from frames, that are centered around specific verbs. This is yet again driven by the dependency parser. The ontology allows for a specification of frames, in the following example:

Er haut ihm auf die Nase. [a]

[a]He punches him on the nose.

In order to extract this interaction, a frame is to be modelled in the ontology. A frame is modelled in the same manner as a relation, but contains more information. An example is shown in figure 4.8. The name of the frame is set to "FAufDieNasehauen" and a couple of children are modelled to this frame. The "ACTION" node contains information about the relations that should be created if a positive match occurs. The ”DA” contains information about the concept that is found in dative position. ”MO” and ”SB” are to model constrains about prepositional phrases (attached to the verb) or constrains about the subject (e.g. it should be a person). The node ”VERB” contains information about the central verb of this frame. From the modelling perspective, this differs from the first stage of the relation extraction. In figure 4.9, an example for a relation is given. A relation comes with an ”ACTION” node, but contains only ”FROM” and ”TO” children which describe constrains about the agents of the binary relation.

The last stage of the relation extraction deals with comparisons (that are introduced by ”als” and ”wie”). If any relation is found to be in a comparison (this is yet again done by the dependency tree), additional relations are created. An example is given, the comparison is highlighted using italics:

Der **Pullover** war **rot** *wie ein Feuer*[a]

[a]The sweater was red like a fire.

One of the previous conditions extracts the color ”red” and attaches it to the ”sweater”. But since there is more information about the color in this snippet, an additional comparison relation between ”red” and ”like a fire” is created automatically.

The relation extraction system is (at the current point in time) capable of correctly dealing with *symmetric* relations and with relations that are *inverse*.

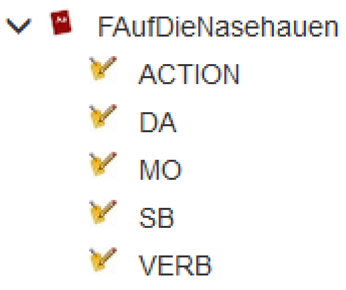


Figure 4.8.: An example frame for OBIE as it is modelled in WebATHEN

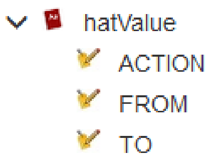


Figure 4.9.: An example relation for OBIE as it is modelled in WebATHEN

During this thesis, the algorithm was applied in the speaker resolution module, in order to extract the explicit speaker for an utterances, as well as to detect, whenever a segment which is enclosed in quotation marks does not depict a speech but a different category such as a thought. At the current point in time, this algorithm comes with about 20 preconfigured ontologies (most usually incomplete) that can be used in order to extract relations about characters. These ontologies comprise optical relations (eyes, face, beard, hair, skin, clothing, ...), social attributes of a character (possessions, social stand), an ontology to extract the age and the health of entities, and ontologies to extract capabilities and habits of entities. Neither of these are complete and no results about their quality are available due to the lack of evaluation data.

Required Extensions to the Algorithm

The algorithm is built upon a compatible format to WebATHEN, so a user friendly editor to build and edit ontologies is available. Even though the format is the same, the way the ontologies are built using WebATHEN differs in almost every aspect, when compared to the intentional way to create ontologies for the medical domain. However so far this has not been an issue and it is not expected to be. In order to increase the compatibility to different ontology editors, (such as Protégé [Gennari et al., 2003] or its web counterpart [Tudorache et al., 2013]) a serialization method into OWL ([McGuinness et al., 2004]) is required.

Additionally, while the extracted relations do offer a possibility to infer additional relations using Meta data (such as reflexive or symmetric relations), there could be an even more detailed support, when it comes to model negations or whether a relation is extracted in a special context (such as a direct speech). This would allow the user to gain alot more control over the algorithm. A further current shortcoming is that the algorithm is not capable of post processing relations. This could include filters which are required for some extractions.

At the current point in time, the extraction process can not store the resulting extractions in an ontology, instead it only produces annotations. This ontology population step is required for the algorithm to obey the aforementioned defintion of an OBIE system.

5. ATHEN

5.1. Introduction

The chapter describes the general purpose annotation environment ATHEN (**A**nnotation and **T**ext **H**ighlighting **EN**vironment) which was developed to manually enrich arbitrary input (text) documents with meta data. ATHEN builds on the philosophy and specification of Apache UIMA, that is meta data is stored in the form of annotations alongside the text. The text itself (commonly referred to as the subject or analysis or short "sofa") is immutable, which means that once data has been loaded (into a *Common Analysis System* (CAS)), only annotations may change but the text is assumed to remain constant.
ATHEN supports many different input and output formats, allows the creation, integration and execution of pipelines using Annotation Engines written in Apache UIMA, both on the local device as well as annotation engines stored on a remote server. The preprocessing tools, that were developed during this thesis are available to be used as a pipeline inside of ATHEN. Since the user can integrate custom typesystems, ATHEN supports the annotation of any type as long as it is conform with the typesystem specification found in UIMA [1]. As a second mechanism to provide ATHEN with a user specified schema for annotation, ATHEN supports the creation and manipulation of OWL-based ontologies, with a dedicated view for medical Ontology Based Information Extraction.
Since 2018, ATHEN has been ported from a desktop application to a web application, named WebATHEN, available for free use without the need of a registration [2]. ATHEN was presented at the DHD 2018 in Cologne [Krug et al., 2018b][3]

This chapter is structured as follows: First the motivation of the creation of a new tool for text annotation as well as its goals is given, followed by an overview of the features integrated in ATHEN in section 5.3. The Use-Cases in which ATHEN was employed are shown in section 5.4. A comparison of ATHEN with previously existing annotation tools in several core aspects is given in section 5.5. This chapter is concluded by some technical details about ATHEN and the reasoning behind internal design choices of ATHEN are introduced and elaborated.

[1]`https://uima.apache.org/d/uimaj-current/references.html`, accessed 20.05.2020
[2]`http://webathen.informatik.uni-wuerzburg.de`
[3]ATHEN was adapted by the project "Redewiedergabe" [Brunner et al., 2019] at the Ids Mannheim.

5.2. Motivation

The philosophy behind ATHEN is to enable the user to define an annotation schema (in the form of an Apache UIMA Typesystem), select documents of choice (no matter if previous meta data was already assigned) and can use ATHEN for a comfortable annotation of those documents using her own schema. ATHEN features specific support for the tasks in the Kallimachos project (namely Character Reference Detection, Coreference Resolution, Relation Detection and Speaker Attribution, see section 5.4). Once the annotation is completed, the results of different annotators could be compared and evaluated in a side-by-side manner, remaining inconsistencies can be fixed and the final document can be exported.

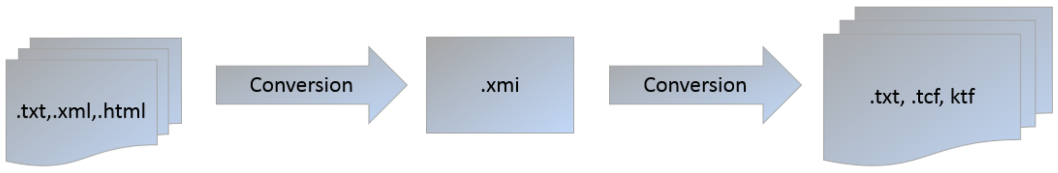


Figure 5.1.: The general workflow with ATHEN. The user can load documents in .txt, .xml (including TEI-XML, .html or .xmi, which get automatically converted into an .xmi file. The user can choose between four output formats after she finished the manual annotation process (.txt, .xmi, .tcf[a] or .ktf (Kallimachos Tab Format, which is similar to the CoNLL tab format[b]))

[a]format is specified here: `https://weblicht.sfs.uni-tuebingen.de/weblichtwiki/index.php/The_TCF_Format`, accessed 20.05.2020

[b]`https://ufal.mff.cuni.cz/conll2009-st/task-description.html`, accessed 20.05.2020

ATHEN itself is designed in the same manner as most programming IDEs, that is at the top-level it consists of different *perspectives* which define their own layouts. The relevant perspective during this work is the Textprocessing perspective, its layout is depicted in figure 5.2

Documents that are currently worked on are shown in the Project Explorer and can be selected and opened to be shown in the editor of ATHEN. The editor displays the text and the meta data in form of the annotations using different, user specified styles and allows for a comfortable overview of the active document. Aggregated information about the current active document can be shown in specialized Views of ATHEN. One such View is to compare annotations to get insight into inter annotator agreements and the purpose of another View is to annotate character references and the coreference information.

If the user places a cursor in the editor, this *selection* affects the Analyzer of ATHEN. This allows to gain a direct overview of all annotations the surround the selection or to view the dependency or constituency parses of a sentence.

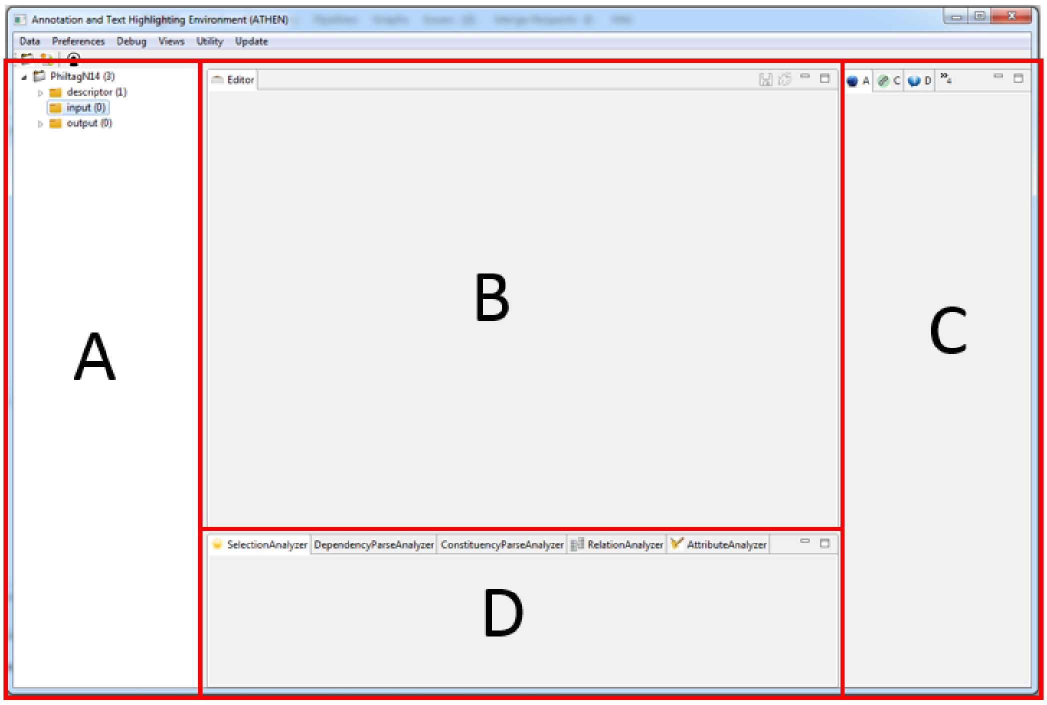


Figure 5.2.: The layout of the textprocessing perspective in ATHEN. Documents in the workspace of the user are shown in the *Project Explorer* A. After double clicking the document gets opened in the *Editor* B and the user can select between different actions by choosing an appropiate *View* C for her task. Information about a specific position in the editor can be gathered using the parts in the *Analyzer* D.

5.3. Features of ATHEN

The section goes briefly over the existing features of ATHEN, more fine grained instructions can be found in the according git-repository[4] and its tutorials in the wiki.

5.3.1. General Purpose Annotation Using the Annotation Browser

One of the main aspects, when deciding whether a given annotation tool is appropriate for a task at hand is whether its schema can be modelled within the annotation tool, and whether the current form of the documents can be used as input into ATHEN. In this chapter, this capability is referred to a general purpose annotation. In order to enable general purpose annotation at least three steps have to be available in a tooling, these are:

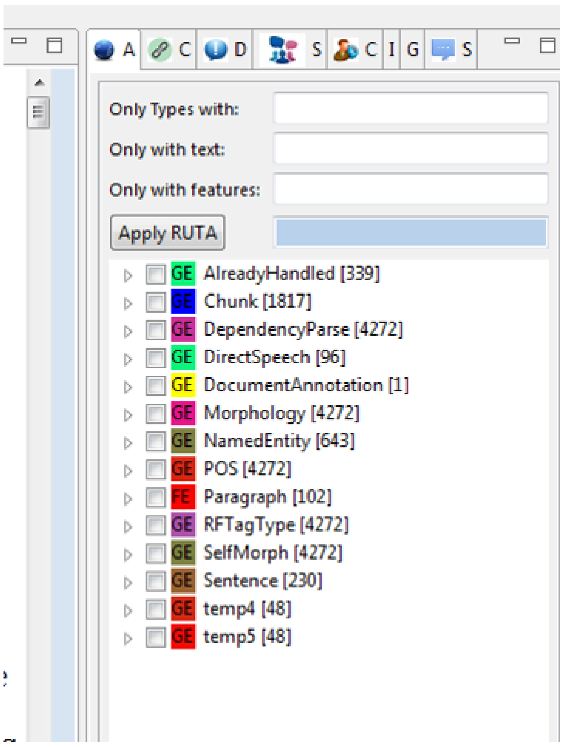
1. Loading an existing document, if possible retaining all of its meta data.

[4]`https://gitlab2.informatik.uni-wuerzburg.de/kallimachos/Athen`

2. Editing the document with a custom schema.
3. Saving the document (in a potentially different format)

ATHEN allows the user to specify a local folder, which in turn is treated as a *workspace* and all actions done inside this folder cause a refresh in the user interface. This way, a user can simply place her documents into the workspace and open them by double clicking the document inside the *Project Explorer* of ATHEN. If no such workspace is wanted, the user can still open documents by simply dragging a document from her file system into the editor of ATHEN.

Defining a custom schema for annotation means to create a valid Apache UIMA Typesystem, which is an *.xml* file and an editor supporting the creation comes with Apache UIMA. A valid typesystem can be loaded into ATHEN which in turn gets merged with the default typesystem integrated into ATHEN. If both - a valid schema as well as documents for annotation are available, the *Annotation Browser View*, depicted in figure 5.3 in ATHEN allows a general purpose annotation (The typesystem which is integrated into ATHEN is depicted in C).

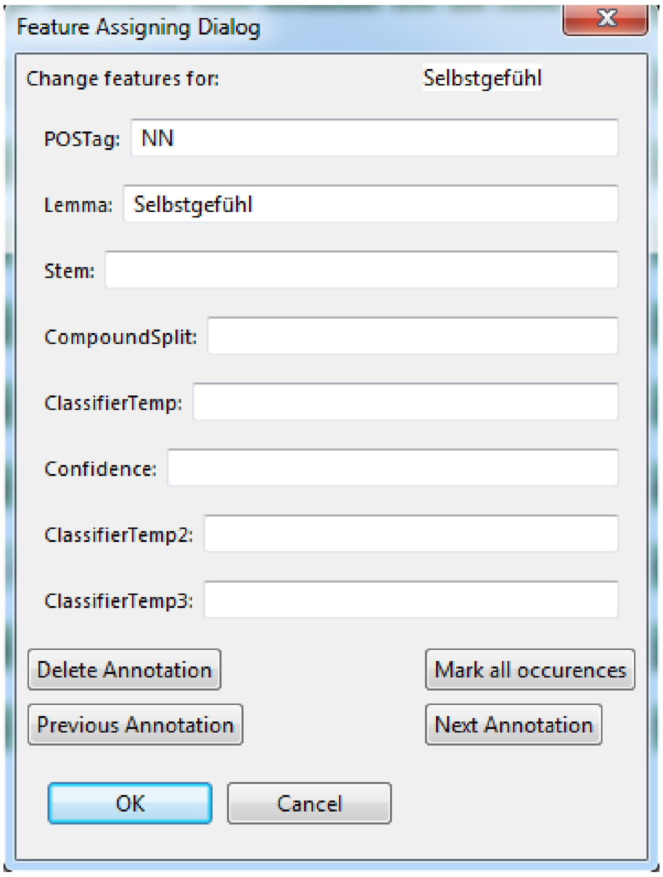


Only Types with:
Only with text:
Only with features:

- GE AlreadyHandled [339]
- GE Chunk [1817]
- GE DependencyParse [4272]
- GE DirectSpeech [96]
- GE DocumentAnnotation [1]
- GE Morphology [4272]
- GE NamedEntity [643]
- GE POS [4272]
- FE Paragraph [102]
- GE RFTagType [4272]
- GE SelfMorph [4272]
- GE Sentence [230]
- GE temp4 [48]
- GE temp5 [48]

Figure 5.3.: Annotationbrowser View, it shows all annotations of different types that are currently available in the document. The selection of a specific instance allows a change of its features and by pressing *delete* the selected annotations can be deleted. A double click on a type creates a new annotation with empty features, based on the current text selection of the editor.

The user can check the types she wants to annotate (which would display them in their according style in the editor). She can then select text in the editor and double click on a given type in the Annotation Browser which will create an annotation of that type without assigning any feature values. Moving the cursor inside a visible annotation will *"select"* it and by pressing *Enter* the user can edit the features of an annotation. The dialog is presented in figure 5.4. Deleting an annotation is simply moving the cursor inside an annotation and pressing *Delete*.


Feature Assigning Dialog

Change features for: Selbstgefühl

POSTag: NN

Lemma: Selbstgefühl

Stem:

CompoundSplit:

ClassifierTemp:

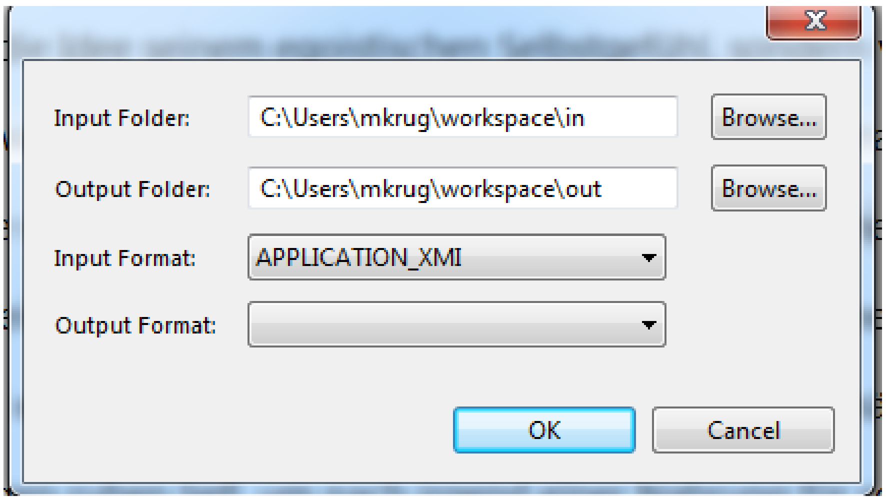
Confidence:

ClassifierTemp2:

ClassifierTemp3:

Figure 5.4.: The dialog to change annotations in ATHEN. It lists all available features and offers a simple input mechanism to update the features. On top it offers a mechanism to navigate between annotations of the same type.

If the user is done editing the document, it can be saved by either pressing the disk symbol in the toolbar of the editor or by pressing ctrl + s. In a subsequent step, the document can be converted into a different format, such as a tab separated format (.ktf) or an xml format (.tcf). As an example, see figure 5.5.

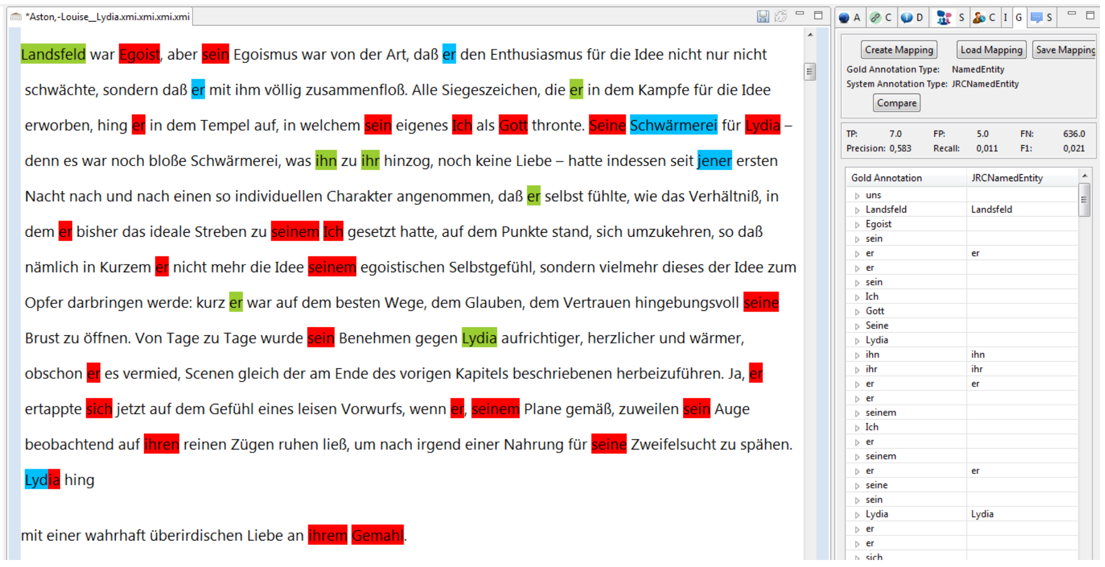


Input Folder: C:\Users\mkrug\workspace\in
Output Folder: C:\Users\mkrug\workspace\out
Input Format: APPLICATION_XMI

Figure 5.5.: Feature conversion dialog, the user can specify an input and an output folder, as well as an according conversion pattern, in this example .xmi documents get converted and the user can choose between the supported output formats. The dialog is availbe via the menu Utility › Convert Documents... .

5.3.2. Comparing Annotations using the Goldstandard View

If different sources of annotations are available (this is usually the case when multiple annotators worked on the same source, or when an algorithm is designed to automatically predict new annotations) then ATHEN supports a comparison among different types using custom rules that hold during comparison, stored in a so called *mapping*. Simply said, a mapping defines when two annotations of a different type are considered equal. The view, shown in figure 5.6, allows the user to either define her own mapping or to load one or multiple mappings she has previously stored on her computer.



*Aston,-Louise__Lydia.xmi.xmi.xmi.xmi

Landsfeld war Egoist, aber sein Egoismus war von der Art, daß er den Enthusiasmus für die Idee nicht nur nicht schwächte, sondern daß er mit ihm völlig zusammenfloß. Alle Siegeszeichen, die er in dem Kampfe für die Idee erworben, hing er in dem Tempel auf, in welchem sein eigenes Ich als Gott thronte. Seine Schwärmerei für Lydia – denn es war noch bloße Schwärmerei, was ihn zu ihr hinzog, noch keine Liebe – hatte indessen seit jener ersten Nacht nach und nach einen so individuellen Charakter angenommen, daß er selbst fühlte, wie das Verhältniß, in dem er bisher das ideale Streben zu seinem Ich gesetzt hatte, auf dem Punkte stand, sich umzukehren, so daß nämlich in Kurzem er nicht mehr die Idee seinem egoistischen Selbstgefühl, sondern vielmehr dieses der Idee zum Opfer darbringen werde: kurz er war auf dem besten Wege, dem Glauben, dem Vertrauen hingebungsvoll seine Brust zu öffnen. Von Tage zu Tage wurde sein Benehmen gegen Lydia aufrichtiger, herzlicher und wärmer, obschon er es vermied, Scenen gleich der am Ende des vorigen Kapitels beschriebenen herbeizuführen. Ja, er ertappte sich jetzt auf dem Gefühl eines leisen Vorwurfs, wenn er, seinem Plane gemäß, zuweilen sein Auge beobachtend auf ihren reinen Zügen ruhen ließ, um nach irgend einer Nahrung für seine Zweifelsucht zu spähen.

Lydia hing

mit einer wahrhaft überirdischen Liebe an ihrem Gemahl.

Gold Annotation Type: NamedEntity
System Annotation Type: JRCNamedEntity

TP:	7.0	FP:	5.0	FN:	636.0
Precision:	0,583	Recall:	0,011	F1:	0,021

Gold Annotation	JRCNamedEntity
uns	
Landsfeld	Landsfeld
Egoist	
sein	
er	er
er	
sein	
Ich	
Gott	
Seine	
Lydia	
ihn	ihn
ihr	ihr
er	er
er	
seinem	
Ich	
er	
seinem	
er	er
seine	
sein	
Lydia	Lydia
er	
er	
sich	

Figure 5.6.: Goldstandard View, given one or more mappings (in this case there is a comparison between a type called *NamedEntity* which is selected to be the gold type against a type called *JRCNamedEntity*. In the editor, all true positives are highlighted in green, all false positives in blue and all false negatives in red. On top, the total amount of them is reported alongside Precision, Recall and F1-score. The table in the view aligns all true positives, and whenever no counterpart to an annotation could be found, then the according cell is left empty. (Any data in this image is just for demonstration and does not resemble any real algorithm).

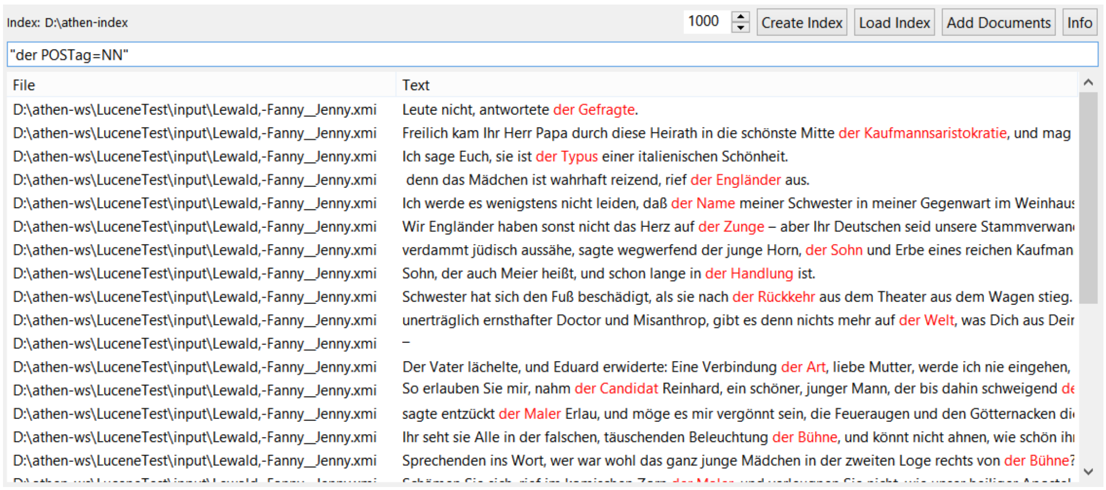
If a single mapping is loaded or created, then the annotation of two types defined in the mapping get compared, if more than two mappings are selected, then it is expected that at least one type is present in every mapping (and treated as *gold* type), and all other types are automatically treated as *system* types and will be compared against the gold type. The mapping itself specifies which type gets considered the gold standard and which type gets compared to this information. By simply swapping the columns in the view, the user can change the current type that is considered to be the gold type (the first column of the table contains the gold type). By clicking the compare button, the user can trigger the automatic comparison as defined by her mappings. The table in the view shows annotations that could be mapped (true positives) those that could not be mapped to any comparing type (false negatives) and those that were detected but are wrong (false positives). On top, among the first two columns, the view calculates and displays the Precision, the Recall and the F1-measure.

5.3.3. Querying Documents and Document Collections using Apache UIMA Ruta and Apache Lucene

Enriching documents with meta data is one core aspects, which is supported by ATHEN, but ATHEN also features two integrated mechanisms to a) pose a query to a single document using Apache UIMA RUTA or b) pose a query to a collection of documents using Apache Lucene[McCandless et al., 2010].

If information about the currently active document is of interest, the annotation browser features the support for simple queries based on the framework Apache UIMA RUTA [Kluegl et al., 2016].

If more involved queries over a document collection are in the focus of analysis, then ATHEN features support for Apache Lucene and its index structures. In order to create an index, the user has to specify an annotation type that is interpreted as a document of the index structure (it could be *DocumentAnnotation* but could also be any other type such as paragraphs or sentences). The next information that is required for creating an index is the type that is used as the basic unit - the token type. If both kinds of information are available, then the user can select all other kinds of annotations and their features that get added as meta data onto the documents and their tokens. This means, that everything that is selected can be queried after the creation of the index. After the index has been created (which might take a while, depending on the size and sheer amount of the documents), it can be loaded at any given time and does not need to be recreated every time ATHEN is started. The query window (see figure 5.7) is then used to query the index and can - at the same time - open the originating documents in the editor for a more detailed inspection.



"der POSTag=NN"

File	Text
D:\athen-ws\LuceneTest\input\Lewald,-Fanny__Jenny.xmi	Leute nicht, antwortete der Gefragte.
D:\athen-ws\LuceneTest\input\Lewald,-Fanny__Jenny.xmi	Freilich kam Ihr Herr Papa durch diese Heirath in die schönste Mitte der Kaufmannsaristokratie, und mag
D:\athen-ws\LuceneTest\input\Lewald,-Fanny__Jenny.xmi	Ich sage Euch, sie ist der Typus einer italienischen Schönheit.
D:\athen-ws\LuceneTest\input\Lewald,-Fanny__Jenny.xmi	denn das Mädchen ist wahrhaft reizend, rief der Engländer aus.
D:\athen-ws\LuceneTest\input\Lewald,-Fanny__Jenny.xmi	Ich werde es wenigstens nicht leiden, daß der Name meiner Schwester in meiner Gegenwart im Weinhau
D:\athen-ws\LuceneTest\input\Lewald,-Fanny__Jenny.xmi	Wir Engländer haben sonst nicht das Herz auf der Zunge – aber Ihr Deutschen seid unsere Stammverwan
D:\athen-ws\LuceneTest\input\Lewald,-Fanny__Jenny.xmi	verdammt jüdisch aussähe, sagte wegwerfend der junge Horn, der Sohn und Erbe eines reichen Kaufman
D:\athen-ws\LuceneTest\input\Lewald,-Fanny__Jenny.xmi	Sohn, der auch Meier heißt, und schon lange in der Handlung ist.
D:\athen-ws\LuceneTest\input\Lewald,-Fanny__Jenny.xmi	Schwester hat sich den Fuß beschädigt, als sie nach der Rückkehr aus dem Theater aus dem Wagen stieg.
D:\athen-ws\LuceneTest\input\Lewald,-Fanny__Jenny.xmi	unerträglich ernsthafter Doctor und Misanthrop, gibt es denn nichts mehr auf der Welt, was Dich aus Dei
D:\athen-ws\LuceneTest\input\Lewald,-Fanny__Jenny.xmi	–
D:\athen-ws\LuceneTest\input\Lewald,-Fanny__Jenny.xmi	Der Vater lächelte, und Eduard erwiderte: Eine Verbindung der Art, liebe Mutter, werde ich nie eingehen,
D:\athen-ws\LuceneTest\input\Lewald,-Fanny__Jenny.xmi	So erlauben Sie mir, nahm der Candidat Reinhard, ein schöner, junger Mann, der bis dahin schweigend de
D:\athen-ws\LuceneTest\input\Lewald,-Fanny__Jenny.xmi	sagte entzückt der Maler Erlau, und möge es mir vergönnt sein, die Feueraugen und den Götternacken di
D:\athen-ws\LuceneTest\input\Lewald,-Fanny__Jenny.xmi	Ihr seht sie Alle in der falschen, täuschenden Beleuchtung der Bühne, und könnt nicht ahnen, wie schön ih
D:\athen-ws\LuceneTest\input\Lewald,-Fanny__Jenny.xmi	Sprechenden ins Wort, wer war wohl das ganz junge Mädchen in der zweiten Loge rechts von der Bühne?

Figure 5.7.: Lucene query window. This image shows the result of a query which extracts all sequences of *"der"* followed by a noun. An excerpt of the results, shown by the original document name as well as the corresponding text snippet is provided and serves as a possibility to examine hypotheses over a document collection.

5.3.4. Preprocessing using Apache UIMA Analysis Engines

Manual annotation can be supported by automatic preprocessing in order to speed up the process. With the modules of the Kallimachos pipeline (see section 2.2) being accessible via an online repository named *Nappi*, ATHEN features the execution of multiple syntactical preprocessing components, as well as the semantic components developed during this thesis.

ATHEN supports a configuration and execution of preconfigured Analysis Engines via the Nappi feature. In the current state, Nappi is realized as an online repository which can be downloaded to the local PC and all contained engines can be loaded, configured and executed in ATHEN. This process is done in the according Nappi-UI which is integrated into the UI of ATHEN and can be accessed by navigating to Utility › Nappi-UI, it is shown in figure 5.8.

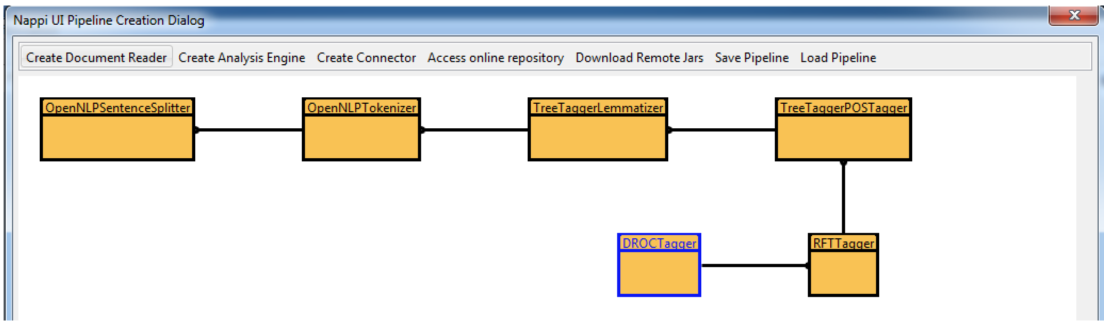
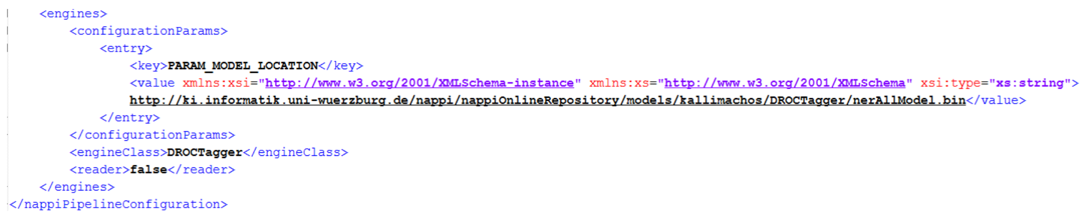


Figure 5.8.: Main view of the Nappi UI, a pipeline can be configured by adding Analysis Engines and connecting them in the according order to obtain a pipeline. By double clicking an Analysis Engine, the user can change the engine and its parameters and save the pipeline once she finished the configuration.

After the creation of a pipeline the user can save a pipeline configuration and execute it via the context menu of the editor. If Analysis Engines are present in a remote repository, then the execution of the pipeline will first attempt to download the according engines and then execute them.

An *Analysis Engine* for Nappi is provided via separate *.jar* files. The jar itself requires a special .xml configuration element that is included. An example snippet for this configuration can be seen in figure 5.9



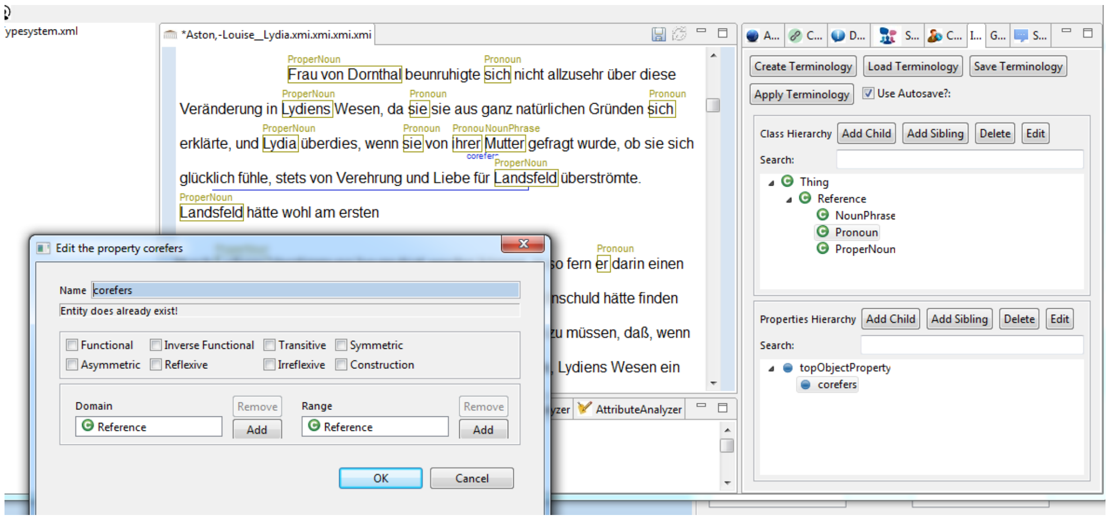
```xml
    <engines>
        <configurationParams>
            <entry>
                <key>PARAM_MODEL_LOCATION</key>
                <value xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xs="http://www.w3.org/2001/XMLSchema" xsi:type="xs:string">
                http://ki.informatik.uni-wuerzburg.de/nappi/nappiOnlineRepository/models/kallimachos/DROCTagger/nerAllModel.bin</value>
            </entry>
        </configurationParams>
        <engineClass>DROCTagger</engineClass>
        <reader>false</reader>
    </engines>
</nappiPipelineConfiguration>
```

Figure 5.9.: An example configuration, in this case for the DROC-Tagger. This component requires solely a model location which is specified via an URL.

5.3.5. OWL Support in ATHEN

If a users goal is to extract information in the form of entities and relations between entities, ATHEN offers support for annotation schemes based on ontologies modelled in OWL. This is very useful if the labeling scheme consists of very detailed hierarchies and relations between concepts of that hierarchy.

ATHEN contains a dedicated and rather potent view if ones goal is to extract information from text. The de facto standard for modelling information extraction is an ontology that is stored in the .owl format [McGuinness et al., 2004]. The *IEView* in ATHEN allows a user to specify and annotate an according schema defined in OWL. The view itself does not support all features available in OWL (for an editor with the full flavors of OWL, see Protégé [Gennari et al., 2003] or its pendant in the web [Tudorache et al., 2013]), and adapts some of the concepts to be more fitting towards information extraction task from texts. A large variety of tasks can utilize a schema that solely consists of a hierarchy of classes (or labels) as well as relations in between those classes. E.g. a coreference annotation might require the classes *proper noun*, *noun phrase* and *pronoun* which could be modelled as children of *reference* as well as the relation *corefers* which is defined between two references, this scenario alongside the annotation of a text snippet can be seen in figure 5.10. After the definition of the classes and relations (called properties) among the classes, one can annotate against this schema. Each annotation that is done this way could be integrated into the ontology and be modelled as an *instance* during the ontology population step.


ypesystem.xml

*Aston,-Louise__Lydia.xmi.xmi.xmi

Frau von Dornthal beunruhigte sich nicht allzusehr über diese
Veränderung in Lydiens Wesen, da sie sie aus ganz natürlichen Gründen sich
erklärte, und Lydia überdies, wenn sie von ihrer Mutter gefragt wurde, ob sie sich
glücklich fühle, stets von Verehrung und Liebe für Landsfeld überströmte.
Landsfeld hätte wohl am ersten

so fern er darin einen
nschuld hätte finden
zu müssen, daß, wenn
Lydiens Wesen ein

Edit the property corefers

Name: corefers
Entity does already exist!

Functional ☐ Inverse Functional ☐ Transitive ☐ Symmetric ☐
Asymmetric ☐ Reflexive ☐ Irreflexive ☐ Construction ☐

Domain: Reference
Range: Reference

Use Autosave?: ✓

Class Hierarchy
- Thing
  - Reference
    - NounPhrase
    - Pronoun
    - ProperNoun

Properties Hierarchy
- topObjectProperty
  - corefers

Figure 5.10.: The ontology view of ATHEN. The view itself (on the right side) features two main blocks. The first block (top right) contains the hierarchy of the ontology in the form of classes that are stored in a tree. The bottom block of the view contains the properties that are defined between classes in the ontology. Each property itself can define a *domain* and a *range* to depict when it is applicable. Properties can be marked to show standard mathematical relation properties, such as *reflexive*, *symmetric* or *transitive*. If classes and properties are defined, the user can proceed to annotate based on this schema.

5.3.6. The ATHEN Editor

This section goes briefly over the concepts behind the main component of ATHEN, the editor, since the editor was programmed from scratch and no good tutorials on how to concretely build such a (rather complicated component) are available. Even though ATHEN is built using SWT, the presented concepts are rather general and it should be able to implement such an editor using any UI framework of choice. For exampe, most of the concepts were reused for the editor that is used in WebATHEN. The following procedure only requires a possibility to draw on the screen as well as a way to measure the extents of strings using an arbitrary font style. In most programming languages this is provided in a so called **Canvas**. The first design choice, when building an interactive component is to whether render it continuously (such as a game, with a given frame rate) or whenever an user event is performed. The editor of ATHEN is based on events in order to refresh its content. The next design choice is the definition of the main data structures. This editor stores and renders its content on the basis of **EditorLines** (see figure 5.11).

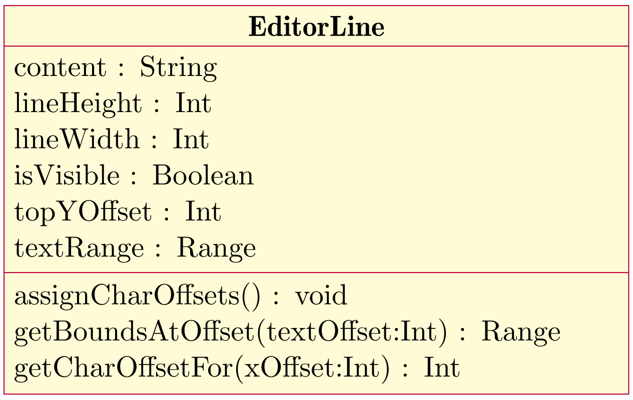


Figure 5.11.: The class EditorLine

Algorithm 2: The function DetermineWrappedLines, which takes text as its input and returns a list of EditorLines, which are subsequently rendered by the editor. It is guaranteed, that all lines do not exceed the width of the editor widget itself.

```
Result: List<EditorLine>
lineOffset← 0;
lineNumber← 0;
while lineOffset<text.size do
    indexOfBreak ← text.indexOf(lineBreak,lineOffset) || text.size;
    lineInFocus ← text.substr(lineOffset,indeOfBreak);
    lineWidth ← measureLine(lineInFocus);
    if lineWidth < widgetWidth then
        create a new EditorLine;
    else
        Greedily Wrap the line at whitespaces
    end
    lineOffset+= lineInFocus.size;
end
```

It stores the textual content of this line in the field **content**, the indices, where the content is found in the full text in the field **textRange**, the width and height of the line in the according fields **lineHeight** and **lineWidth**. The field **topYOffset** represents the top pixel of the users screen where the line is to be drawn or -1 if the line is invisible. The most important methods are **assignCharOffsets**, which is called whenever coordinates for a character are requested for the first time. Its job is to derive a bounding box for every character of the content and to store it in an according data structure. This assures that subsequent calls can be served rather quick. More precisely there are two main ways to query a line, the first is by calling the method **getCharOffsetFor**, which returns the index of the character (the position based on the whole document) that resides in the current line at pixel *xOffset*. This method is repeatedly called, whenever the mouse is moved in the editor and therefore requires to be rather efficient (in this case a dictionary

lookup). The second method **getBoundsAtOffset** returns the *Range* for the character at a given offset. These **EditorLines** are initially created (using algorithm 2), whenever a new document is loaded into the editor, and they are recreated, whenever the editor or its window were resized.
With created lines, the editor can now proceed to render these lines on the display. This was an important decision, if all lines are rendered at once and the user can scroll between them, then the editor gets slow even with a moderate amount of lines. This is why the editor is tracking the top line that is currently visible on the screen of the user and only renders subsequent lines until the screen of the user is filled. This means that the editor itself never has a size larger than the visible size of the editor. The possibility to scroll between lines had to be programmed manually as well. The scrollbar is basically just a fake indicator which simulates the total size all lines would require if they would be drawn. Changing the scrollbar (by mouse scroll or some keyboard inputs) only changes the top line that is currently visible and renders this portion. Using this method, even documents with millions of lines can be rendered without any issues. The next important operation is how to model and render the annotations in the editor. Due to the standoff markup of Apache UIMA, every annotation carries a **begin** and an **end**. These features allow for an easy sorting so that only these annotations can be filtered that reside in the currently visible area on the users screen. This list of visible annotations has to be updated, whenever new annotations are added/removed or changed, whenever the lines are changed and whenever the selection of the visible lines is changed. This is implemented using a binary search in a sorted array and can therefore be considered rather fast. The editor stores **styles** in the form of **drawingStrategies** for each of the visible annotations (visible annotations are either defined by the user or by the current active view of ATHEN). Such a drawingStrategy is a function that gets called for every annotation of the assigned type and the programmer can now draw arbitrarily into the user interface of the editor. ATHEN comes with about 10 general drawingStrategies (Box, Border, Background, Underlined, ..) and about further 10 special drawingStrategies that are used in the different Views.

Aside from having the annotations sorted by begin and end offsets, the editor represents different layers for different annotation types. Since the annotations are rendered in their according drawingStrategies, this allows for control as of which annotations should be drawn on top of other annotations. One additional and rather crucial component for a good performance of the editor is the use of so called **double buffered rendering**. Since every (visible) annotation requires additional calculations in order to be rendered on the screen, it is smart to render the entire update of the editor into an image (invisible to the user) and once all calculations are finished, the image is drawn entirely into the editors widget. This ensures a smooth transition between frames. After all annotations have been rendered, the editor is rendering the text selection (so that it is drawn on top of all other annotations), if such a selection is available. The editor then defines a list of events where different components can register for to be notified. This forms the backbone of the communication between different parts in ATHEN. Even though this allows for a powerful editor, I will still note its limitations:

- All lines and graphics that are rendered by a drawingStrategy do not respond to user input (since they are just colored pixels)
- No collision detection and resolution between overlapping styles (This means that some strings are impossible to read as they are drawn on top of each other)
- Requires a static text, a hard constraint induced by UIMA
- At the current state, drawingStrategies can not make use of different Fonts for the styling of annotations. Since a different font would introduce different line breaks, but the EditorLines are only determined using the raw text.

Aside from the third aspect, all limitations can be resolved, however especially the second point requires a carefully tuned algorithm in order to ensure fast rendering times (in the terms of less than 25ms for the user to not recognize any lag, e.g. when marking text).

5.4. Use Cases of ATHEN

ATHEN was used for the annotation of several different tasks during the Kallimachos project. This section goes over the dedicated functionality for the tasks relevant for the annotation. Starting with the creation of the DROC corpus, which contains character references, coreferences in between the character references as well as direct speech utterances alongside their speaker and addressees. Furthermore ATHEN was used to annotate a (small) data set for the evaluation of nominal and prepositional phrase for the evaluation of syntax parsers. For the task of relation extraction, a data set comprising 213 expert summaries of novels were annotated with character references, coreferences and social as well as family relations. In the same manner, we annotated about 7000 interactions between character references into about half of the documents available in DROC. The CliGS-group [5] used ATHEN to quickly label direct speech annotations as well as their introducing frames in french historical novels [Schöch et al., 2016].
This next section goes over the dedicated support in ATHEN for the aforementioned tasks.

5.4.1. Annotation of Character References and Coreferences

ATHEN was initially designed as an environment to quickly annotate character references and coreferences but developed towards a general purpose annotation environment based on Apache UIMA. The *Coreference View* in ATHEN, depicted in figure 6.3, was the core feature of ATHEN. The view is built into two parts. The top part contains all necessary buttons to quickly annotate or change existing annotations, as well as the possibility to apply the preprocessing engine. The bottom part contains the *entity table*, which contains all references that are marked in the text and offers some filter abilities.
The workflow of this view is as follows:

[5] `https://cligs.hypotheses.org/`, accessed 20.05.2020

1. Invoke the automatic detection of CR
2. Start at the beginning of the text, confirm, delete or add new annotations
3. Once finished adjust the gender, number and type information in the table

These three steps guarantee that a single pass over the text is enough to label both, the spans and the according coreferential information, and therefore utilizes the time spent very efficiently. During the Kallimachos project we decided to only annotate entities by providing a unique identifier and reusing this identifier at every reference. The main advantage of this process is that the user does not need to spend her time to find the previous antecedent, which is sometimes pages away from the current reference. Annotating by using an Identifier also allows an additional feature to speed up the assignment. By selecting a reference (or highlighting some text in the editor), the user can double click on any entry in the table with the correct ID and the information gets carried over to the selected span. Once the user is done marking all spans and assigning the correct id's, she can activate the *ID-filter* in the table to only show the very first instance of every entity. She can then continue by adjusting the type (*"Core", "AppTdfW", "AppA" or "Pron"*), as well as the gender and number. If the annotator is not secure with her choices, she can set a flag to mark this instance as *"uncertain"*.

Following this process, DROC and the Kindler data set were annotated.

5.4.2. Annotation of Direct Speech Utterances, Speaker and Addressees

Since the automatic attribution of speaker and addressees onto direct speech utterances form a key aspect of this work, a dedicated view for this annotation was created (see figure 7.3. This view specializes on the annotation of text snippets that represent direct speech utterances. The user can proceed to mark text in the editor and click on **New Direct Speech** in order to create a new utterance. This utterance will now be highlighted in the editor but no internal features will be assigned to the annotation. In order to enable the annotation of different types of speech, every utterance comes with a **Category**, a string that can be assigned by the user. During the annotation of DROC, not only direct speeches were marked but also *thoughts*, *citations* or *writings*. For every utterance, multiple so called *contexts* can be assigned. A context can for example be used to represent a frame for a speech. This functionality was used in the CLiGS group in order to mark utterances in French novels in order to develop automatic components later on.

The main purpose of this view is to assign speaker and addressee to a direct speech utterance. A speaker and addressee is an annotation of the type **NamedEntity**, which means that in order to annotate a speaker/addressee, the according reference has to be available. In the table of the view, a list of all references of the document (or whenever the Frame filter is enabled just those that are visible on the screen) is shown and the user has to find the according reference in the list. A double click in the according column of the table assigns the reference onto the utterance and subsequently gets highlighted in the editor as long as the according direct speech is selected. This view was used for the annotation of more than 4000 direct speech utterances as well as the assignment of their speaker/addressees during the annotation of DROC.

5.4.3. Annotation of Syntactical Information in ATHEN

ATHEN supports both of the major linguistic theories for the representation of syntax, annotation using **Dependencies** as well as the annotation using **Constituencies**. For both theories, dedicated *Analyzer* parts have been developed for ATHEN. An analyzer reacts to the current selection of the editor. In the case of the **DependencyParseAnalyzer** and the **ConstituencyParseAnalyzer** this means, that whenever a new sentence is clicked in the editor, the content of the analyzer changes. Both components found use when selecting the appropriate preprocessing components for German historic novels. For the case of annotated data for constituency based syntax trees, about 800 complex sentences (as selected by an active learning procedure) from the novels and about 200 segments of the medical domain *körperliche Untersuchung (physical examination)* have been annotated in this manner. On top of pure annotation potential of both components, they can also be used to explain mistakes in later stages and therefore serve as a debugging mechanism.

This section explains the core elements of both components, starting with the DependencyParseAnalyzer.

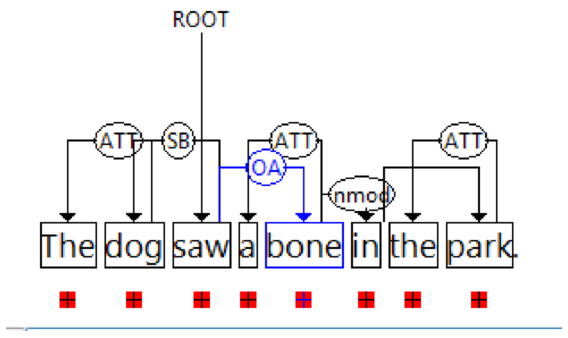


Figure 5.12.: An example sentence, fully annotated with dependency annotations in the DependencyParseAnalyzer.

DependencyParseAnalyzer: The DependencyParseAnalyzer shows a dependency graph as represented in figure 5.12. Each dependency relation is represented as a directed edge, where the dependent is drawn at the end of the edge and the head at the beginning of the edge. Changing an edge means that the user has to select a token (the selected token is drawn with a blue box) and then hovering to the red box underneath the head token. A click on this red box then changes the edge. By pressing *Enter*, the dependecy relation of the current selected edge can be adjusted. If a token resolves to *root*, then a click onto its own red box will create an edge to the virtual root node. Since for sentences with many tokens, the view will get confusing rather fast, once a single token is selected, then only the edges that involve this node are drawn, clicking anywhere outside of a token in the view will yet again switch to the mode in which all edges are drawn. Changes that are made in this analyzer are transferred live into the according annotations of the editor, so that reselecting the same sentence should contain previous changes.

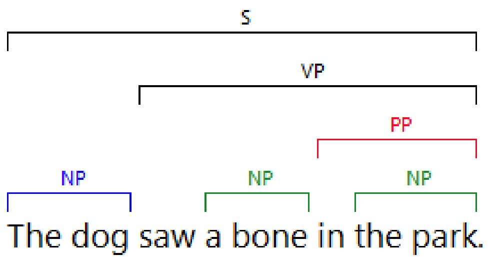


Figure 5.13.: An example sentence, with constituency annotations in the ConstituencyParseAnalyzer.

ConstituencyParseAnalyzer: The ConstituencyParseAnalyzer allows the annotation of classical phrase based syntax trees (see figure 5.13). This is done by selecting the according text that is to be put into a phrase and by pressing *Enter*. This will create an empty bracket. These brackets can then be selected by clicking on them and the label (and other features, such as the semantic component) can be edited. The hierarchy is inferred automatically since larger phrases automatically depict the parent of the underlying phrases. Since the most used labels for phrases are *NP (noun phrase) or prepositional phrases (PP)* the analyzer features a keyboard shortcut to create a bracket and immediately assign the label. In the same manner as the DependencyParseAnalyzer all changes are immediately transferred to the main document.

5.4.4. Annotation of Relations and Interactions

ATHEN was used to annotate two data sets containing relations. Relations, based on the schema depicted in section 4.3.1 were annotated into the expert summaries of Kindler, in total about 6600. Interactions, using the guidelines presented in section 4.6 were directly annotated into 40 documents of DROC. In total about 7500 interactions have been labelled with ATHEN.

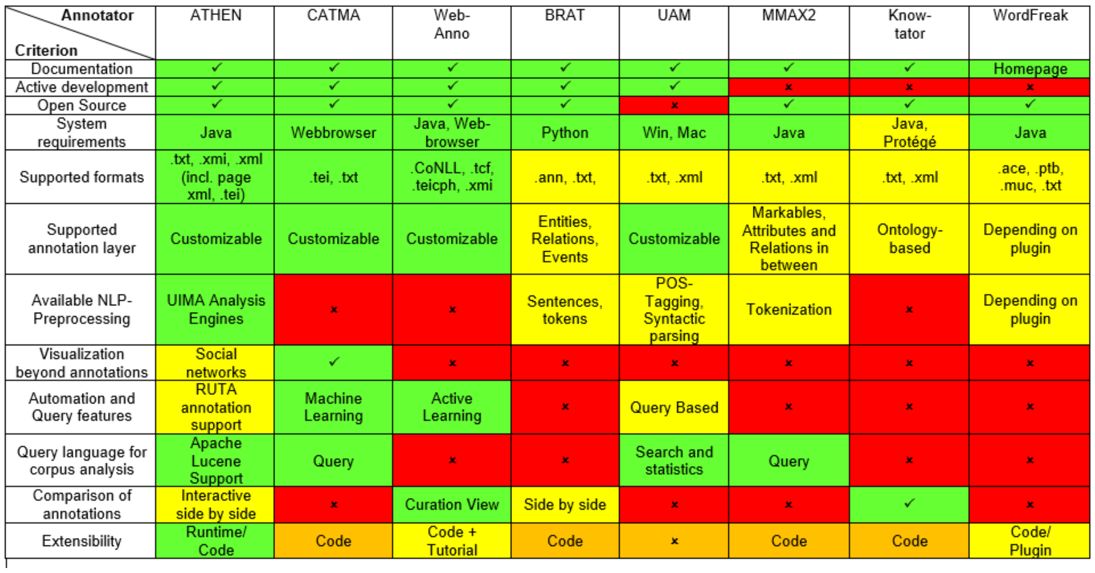
Creating an interaction or relation can either be done in a general way using the OWLView (presented in section 5.3.5) or by using the combination of the **RelationAnalyzer** and the **CoreferenceView**, however this is only useful if the relations are formed between character references (as it is the case for the relations in Kindler or interactions). A relation can be created by selecting text in the editor and by clicking **Add Relation** in the CoreferenceView. This creates an empty relation. Selecting this empty relation in the editor activates the RelationAnalyzer and allows the user to define both agents of the relation, as well as the label for the relation.

5.5. Comparison of ATHEN to existing Tools for Text Annotation

ATHEN is compared to three web-based and four desktop applications in 12 categories by adapting most criteria defined by Neves and Leser [Neves and Leser, 2012] to compare different annotation tools:

1. Availability and up-to-dateness of the documentation
2. Active development at the present time
3. Source code for download
4. Complexity of system requirements
5. Interoperability by supporting certain formats
6. Support of different annotation layers (e.g. POS-tagging, parsing or relations)
7. Support of NLP-preprocessing to speed up manual annotation
8. Support of visualization beyond the display features of the editor
9. Support of self-learning systems to speed up manual annotation
10. Support of querying annotated data
11. Possibility for an inter-annotator-agreement, this is important for projects, in which more than one annotator labels the same documents
12. Extensibility, can the application be modified at runtime or does it need a rebuilt of the entire system.

We explicitly do not want to compare subjective features like usability or how the annotations are presented. The aggregated results are depicted in figure 5.14



Annotator / Criterion	ATHEN	CATMA	Web-Anno	BRAT	UAM	MMAX2	Know-tator	WordFreak
Documentation	✓	✓	✓	✓	✓	✓	✓	Homepage
Active development	✓	✓	✓	✓	✓	×	×	×
Open Source	✓	✓	✓	✓	×	✓	✓	✓
System requirements	Java	Webbrowser	Java, Web-browser	Python	Win, Mac	Java	Java, Protégé	Java
Supported formats	.txt, .xmi, .xml (incl. page xml, .tei)	.tei, .txt	.CoNLL, .tcf, .teicph, .xmi	.ann, .txt,	.txt, .xml	.txt, .xml	.txt, .xml	.ace, .ptb, .muc, .txt
Supported annotation layer	Customizable	Customizable	Customizable	Entities, Relations, Events	Customizable	Markables, Attributes and Relations in between	Ontology-based	Depending on plugin
Available NLP-Preprocessing	UIMA Analysis Engines	×	×	Sentences, tokens	POS-Tagging, Syntactic parsing	Tokenization	×	Depending on plugin
Visualization beyond annotations	Social networks	✓	×	×	×	×	×	×
Automation and Query features	RUTA annotation support	Machine Learning	Active Learning	×	Query Based	×	×	×
Query language for corpus analysis	Apache Lucene Support	Query	×	×	Search and statistics	Query	×	×
Comparison of annotations	Interactive side by side	×	Curation View	Side by side	×	×	✓	×
Extensibility	Runtime/ Code	Code	Code + Tutorial	Code	×	Code	Code	Code/ Plugin

Figure 5.14.: Comparison of ATHEN in 12 categories against seven other systems. No tools excels in all categories but ATHEN is comparable on most categories and excels in the preprocessing category.

All of the listed tools have an accessible documentation, either web-based or as a PDF, available for download. Besides UAM [O'Donnell, 2008], every other application is listed as open source, so at least extensions based on code level can be made. WebAnno [de Castilho et al., 2016], which can be considered as an extension to the BRAT system [Stenetorp et al., 2012] is the only application having a tutorial supporting a new developer to make changes in their project. ATHEN stands out in the sense that extensions to its UI can be made at runtime, therefore easing the process of adding functionality to it. WebAnno supports the largest number of formats and comes with an active learning component which is thought to speed up annotation, however lacks integrated NLP-preprocessing which ATHEN is capable of. CATMA [Meister et al., 2017] is the only project alongside ATHEN, which offer a way to visualize aggregated data that result from the execution of user queries. ATHEN is the only tool, capable of displaying character networks on the fly. Being a standalone web application, CATMA itself does not support NLP-preprocessing. ATHEN comes with the support of the execution of UIMA analysis engines, accessible from web repositories or a local repository, giving the user a chance to integrate her custom-made annotators. Four tools, ATHEN, UAM, MMAX2 [Müller and Strube, 2006] and CATMA feature an integrated query language which helps to analyze existing corpora. Most tools allow the annotation of user-defined annotation schemas earning therefore the title "generic annotation tool". Alongside UAM, ATHEN supports the annotation based on queries, while UAM defines its own language, ATHEN supports the annotation using Apache UIMA Ruta [Kluegl et al., 2016] rules for queries of the document which is currently active, as well as queries using Apache Lucent for entire collections of documents. Three of the listed tools, MMAX2, Knowtator [Ogren, 2006] and WordFreak [Morton and LaCivita, 2003] are currently no longer in active development.

5.6. Technical Details of ATHEN

This section describes the general structure of ATHEN, the ideas for extensibility and the frameworks that are used for the development. ATHEN is designed to be a desktop application, which runs on Windows, Linux and MacOS Operating systems.

5.6.1. Frameworks

In its core, ATHEN builds upon the framework *Open Services Gateway initiative* (OSGI) [Alliance, 2003] which acts as a layered model and enables communication of different projects (called *bundles*) in a single application. OSGI allows the user to start an application by configuring a set of bundles that need to be launched at startup in a given order, and on top unload or reload any bundle if problems exist as well as include new bundles into existing running applications via extension points. An overview over the layered model can be seen in figure 5.15

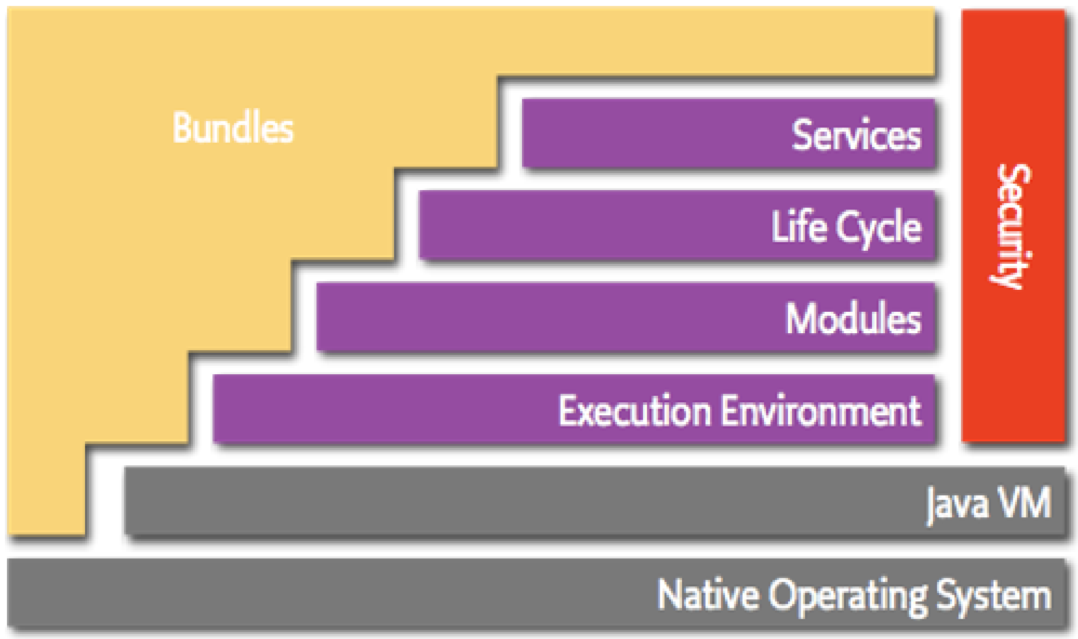


Figure 5.15.: The OSGI layered model, an application consists of a set of projects (bundles) which intervene with the users computer on different levels that range from native operating system calls to JVM calls to the workflow integrated in the own programm. Image taken from the official website[a]

[a]`https://www.osgi.org/developer/architecture/`, accessed 20.05.2020

The Eclipse Rich Client Platform (RCP) [McAffer et al., 2010] is built upon OSGI and therefore was used for the development of ATHEN. Eclipse RCP development is well integrated into the Eclipse platform (which is unsurprisingly since the IDE itself is an RCP application) and comes with a vast set of tooling for easing up the complicated configuration. By the usage of Eclipse Tycho [6] a headless build using Apache Maven [Miller et al., 2010] is made possible. Even though Eclipse RCP supports all three major Java GUI Frameworks (AWT/Swing [Loy et al., 2002], SWT [Merks et al., 2003], and JavaFX [Clarke et al., 2009]), the usage of SWT is promoted and the majority of ATHEN is built in SWT. Executables created by Eclipse RCP are platform specific (since OSGI has platform specific bundles) which means, the user has to download and use a version suitable for her system. The only other requirement is a correct installation of Java with a version of at least Java 8.

The second major framework present in ATHEN is Apache UIMA. Every document gets converted into a so called *Common Analysis System* (CAS) and the user changes and adds meta data in the form of annotations. With UIMA in its core this enables the integration and execution of *pipelines*, automatic text processing components (called *Analysis Engines* able to enrich the incoming CAS with additional meta data that are stacked and executed one after another. Apache UIMA defines the meta data in the form of annotations, which in turn are defined in a dedicated *Typesystem*, an *xml* document describing the types and its attributes.

5.6.2. Adaptability and Extension Points

ATHEN features a variety of so called extension points which serve the purpose of integrating additional components that can even be loaded into a running instance of ATHEN. At the current state, ATHEN supports loading additional views on the fly as well as an integration of additional NLP-preprocessing engines if they are implemented with the according UIMA specification for Analysis Engines. The latter case, the integration of additional NLP-preprocessing engines is depicted in more detail.

The submodule in ATHEN that is responsible for the integration, configuration and execution of NLP pipelines is called Nappi (**NA**tural Language **P**re**P**rocess**i**ng"). It consists of an User Interface which is integrated into ATHEN (shown in figure 5.8), a local repository as well as a remote repository [7]. Both repositories contain previously compiled versions of Apache UIMA Analysis Engines - each one as a single, separate project - enriched with meta data, so that ATHEN can include those projects into a running instance.

[6]`https://www.eclipse.org/tycho/`, accessed 20.05.2020

[7]The remote respository with previously compiled Analysis Engines of the Kallimachos project can be accessed here: `http://ki.informatik.uni-wuerzburg.de/nappi/nappiOnlineRepository`

5.7. History of ATHEN and Review of Design Choices

Since no software can claim to be designed flawless, this section reviews the design choices under the aspect of state-of-the-art software development schemes and frameworks. When I started to work at the Chair for Artificial Intelligence, the main framework for user interface development was by extending the Eclipse framework using the Eclipse RCP-3 platform. Since most competence about this platform was available as well as the fact the there was an annotation editor (provided by Apache UIMA) that could be reused heavily influenced my decision of the framework towards Eclipse-RCP 3 as well. With more and more features to be incorporated into the CoreferenceView, the editor of Apache UIMA did not offer enough capabilities, and on top it does not scale to large documents (such as novels). The first version of the editor (see section 5.3.6) was then created using the RCP-3 framework. Developing software using this platform comes with a heavy downside, namely that the software cannot be delivered as a standalone software, it is always dependant on Eclipse and merely acts as a plugin into eclipse. Developing software in this manner will naturally always generate huge applications since an entire Eclipse is required as its dependency. With the release of Eclipse RCP 4 in 2013, a new generation to develop software using the components of Eclipse was released. ATHEN was ported to RCP-4 in late 2015, with more documentation being available, especially recommending the tutorials provided by Vogella[8]. The transfer from RCP-3 to RCP-4 felt very motivating since it allowed ATHEN to be its own application for the first time. The philosophy of RCP applications, to create small software plugin projects that can be reused very flexible, introduces some overhead when creating the project, but pays off in the long term, since the code is structured in a very clear manner - having one project for every component of ATHEN. The mixture of declarative development (e.g. the top layout of ATHEN is defined in a declarative way, using dedicated tool support) and actual programming of the individual components using very clear and intelligent APIs of SWT ensured a pleasant way of development. I especially want to promote the object oriented concept *Inversion of control*, which not only provides more flexibility to the user (compared to abstract classes and interfaces), but also allows for a much easier way of testing individual components.

But Eclipse RCP-4 does not only come with positive aspects, in fact there are a multitude of downsides. The first one is the way dependencies to other projects or libraries are handled. This can either be done using Apache Felix [Gédéon, 2010] or by using Eclipse Tycho, both being very complicated to work with. Since Tycho is the recommended way to handle dependencies as well as building the application I will go into its main downsides:

1. The configuration requires a complex composite of different modules. A module consists of features which in turn consists of bundles (which are modules themselves)

2. It does not allow the integration of third party dependencies in the classical Maven

[8]`https://www.vogella.com/tutorials/EclipseRCP/article.html`, accessed 20.05.2020

way, instead the user has to convert the Maven dependency into a compatible Tycho bundle and host this bundle in an update site (which the user has to set up herself). This is a very heavy downside since adding new dependencies becomes a lot of work, while in the usual Maven way, this is just the addition of some lines into the corresponding .pom file.

The next downside of Eclipse RCP-4 is that its main way of creating user interfaces is SWT. Even though AWT and JavaFX are supported as well, the bundle which grants interoperability is very unstable and introduced more problems than it solved, resulting in only a small amount of features that can be used out of the box without running into issues, which are sometimes very hard to resolve. As already mentioned, the API of SWT is great, but its main appeal is also its main downside. It is built to reuse the widgets of the underlying operating system, creating a very natural *look and feel* for the user. The truth however is that there is no guarantee that code which works fine on one platform actually works on another platform, resulting in software that is extremely unpredictable. This last issue is extremely frustrating to work with, since it forces you to use workarounds so that your code does work on different platforms - **if at all!**.

How to develop a modern application? If it has to be a desktop application and uses Java, the best way to develop an application is by the use of JavaFX. But current trends show, that the need for desktop applications is declining, and the development of web-apps which imitate the feeling of a desktop application are the norm (e.g. see chat clients such as the client of Skype or Discord, which are basically making use of the Electron framework[9], which comes with a built-in web browser (in these cases, Chromium [Barth et al., 2008])). Developing a web app instead of a desktop app has many benefits. Not only can the application be used as both - web app and desktop application, but the user interface can be created and styled using HTML and CSS. Since HTML5 and CSS3, the creation of responsible, beautiful and dynamic user interfaces can be done entirely declarative, with an abundance amount of tutorials available on the web and a large amount of available jobs in this sector (at the time of writing). The downside of web development is that it is very hard to find and use the current state-of-the-art of web development, since so many frameworks appear on a regular basis. The aspects, that almost all devices feature a Browser and modern development of User interfaces using HTML, CSS and Javascript has become easier, I would suggest to always consider the creation of a web-app at first glance and only rely on Desktop application if there are hard criteria against a web-app.

[9]`http://electronjs.org/`, accessed 20.05.2020

6. Character Reference Detection

6.1. Introduction

[1] Detecting the main characters in literary fiction is a crucial part for gaining insight into the content of a text, e.g. by using character networks, where usually the characters are modelled as nodes and their interactions are depicted as edges. A second use for character reference detection is to detect parts of the plot, since the plot is usually a sequence of actions of the main characters.

We denote the process of marking any text snippet that refers to a character of the input text as character reference detection (CRD). This work deals with the German phenomenon of finding all these spans in historical novels. The characteristics of this task are in between classical Named Entity Recognition (NER), where only proper nouns are detected, and Coreference Resolution (CR), where all noun phrases are detected that refer to a real world or fictional entity. The task in this chapter is a mixture of those two steps: On the one hand every named entity which refers to a character of the novels has to be found, but on the other hand, all other noun phrases that refer to characters in the text have to be detected as well. Similarly to the mention detection step of typical CR systems a character reference can appear as either a:

- Name (e.g. ”Effi”, which we call ”core” references)
- Noun Phrase (which we call appellatives) (”Gärtner” [Gardener])
- Pronoun (”er”, ”sie”, ”ihre”, ”seine”, ... [he, she, her, his, ...])

Over the course of the last 15 years, publicly available, annotated corpora for both tasks have been released for the German language, however none of them contains literary texts and usually the snippets that form those corpora are very short, for example newspaper articles or even just sentences extracted from Wikipedia, which have been labelled manually.

It is a common situation that existing tools were originally designed for a different domain. In order to get findings of a new domain, one has to find a way to either overcome this shift or to adapt existing tools. To complicate things further, previously released tools for NER, such as Stanford NER [Faruqui et al., 2010], are only trained to detect names and do not find appellatives or pronouns which refer to characters. The problem at hand combines both challenges, a shift in domain from newspaper and Wikipedia articles to literary texts and a change in the task from finding solely names to finding all names, appellatives and pronouns that refer to characters in a literary text.

[1] A large portion of this chapter was submitted for publication [Krug et al., 2019a].

Depending on ones goals it might suffice to adapt existing tools and therefore avoid the problem of explicitly re-annotating an entire corpus, which is not only costly but also time consuming. We provide insight into four different possibilities to build a CRD, starting with previously available resources for the German language. These four different routes, depicted in figure 6.1, show entirely different characteristics.

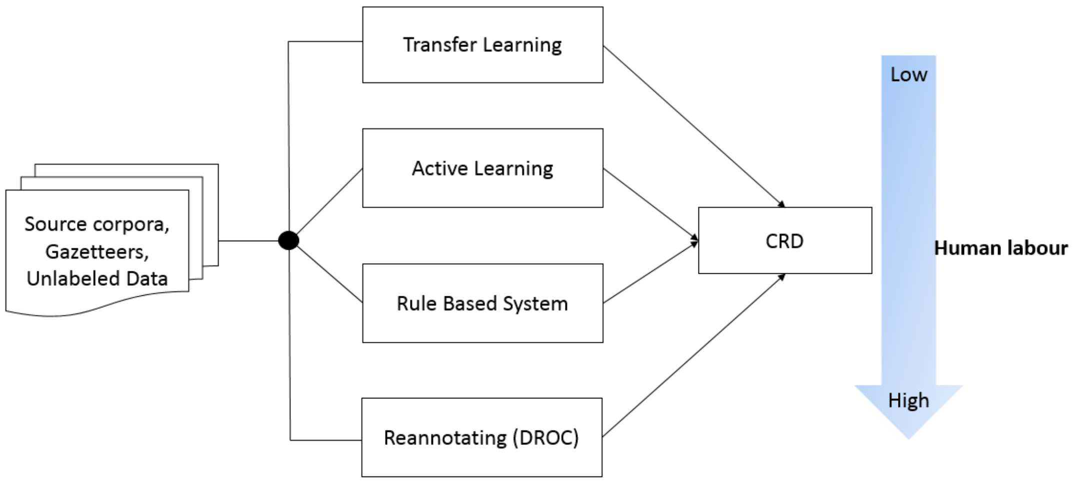


Figure 6.1.: In this work, we experimented with four ways to obtain a character reference detection (CRD) system, beginning with previously released corpora (CoNLL and GermEval), gazetteers as well as unlabeled data of the literary domain. The easiest way in terms of human labour is to use transfer learning and adapt existing tools (in our case the Stanford NER component). Another way to get a component for the target domain is by annotating only instances proposed by Active Learning. The third way is to tackle this task using a rule-based system, which is dependent on the amount of time and capabilities of the rule engineer as well as the quality of the existing algorithms used for preprocessing. The fourth, without question the most laborious, is annotating documents in the target domain using the according task specification and training a classifier in a supervised manner. However this approach is also the most promising in terms of quality.

The first one is to start with existing resources (such as the corpora labeled in CoNLL 2002) and create a system that is able to compensate the shift in domain (and in our case also a shift in the task). This is discussed in section 6.4.1 where we experimented with techniques from *Semi-Supervised Learning*, especially *Transfer Learning*. The second route is to label only as many instances as necessary for a required quality, more specifically we employed techniques found in *Active Learning* to train our classifier. The third route is to express knowledge about the domain and about the task in the form of a rule-based system, which has the advantage to not rely on training data but its quality is highly dependant on the capabilities of the rule engineer. The fourth route is to treat this task as an entirely supervised task on a new domain, which includes the creation of a corpus and supervised training of classifiers specifically for this task.

This chapter is structured as follows. First an overview of existing methods for these tasks is given in section 6.2. In section 6.3, we present the data set we used to compare the performance of the different methods as well as the preprocessing applied to the texts. This is followed by four sections, each representing one approach: Transfer Learning and Domain Adaptation in section 6.4, Active Learning in section 6.5, our rule-based system in section 6.6. Section 6.7 shows the results that can be obtained by using state of the art machine learning approaches in a supervised setting. The combination of our rule-based system with the supervised classifier is presented in section 6.8. This is followed by an error analysis of the different systems in section 6.9 and an analysis of the generalization capabilities to a different data set in section 6.10. Finally, in section 6.11 a discussion of the results as well as the strengths and weaknesses of the different approaches is given.

The contributions of this work can be summarized as follows:

- We provide guidelines for researchers, who face the same challenges that we did - having research ideas in a domain without annotated data.
- We outline the strengths and weaknesses of the individual approaches with respect to handling a shift in both domain and task.
- We release our algorithms for public reuse in the according git repository[2], which on top of the CRD system, offers to date the most extensive tool set of text processing for German literary texts.

6.2. Related Work

Character reference detection can neither be mapped to exactly fit the definition of NER nor the task of mention detection but can better be understood as a processing step residing that is contained in the task of mention detection and overlaps the task of Named Entity Recognition (see figure 6.2.

This section outlines the dominant approaches to NER since this is what is more prevalent in this chapter.

For NER an interesting development can be observed. Early approaches made use of hand crafted rules based on rule templates [Volk and Clematide, 2001] or finite state transducers [Didakowski et al., 2007]. Until CoNLL 2003 these approaches were still rather wide spread. Since then, training data which heavily favours machine learning (ML) approaches was available. Successful rule-based systems, as opposed to systems based on machine learning, treat the problem in multiple phases from different perspectives (e.g. they detect names in prominent spots, and propagate the detected names through the document(s), where the first step operates local and the second operates on a global scale). The data released by CoNLL consists of very small documents, weakening the effect of rules that act on a global scale. With the data from GermEval 2014 this

[2]`https://gitlab2.informatik.uni-wuerzburg.de/kallimachos/CoreferenceResolution`

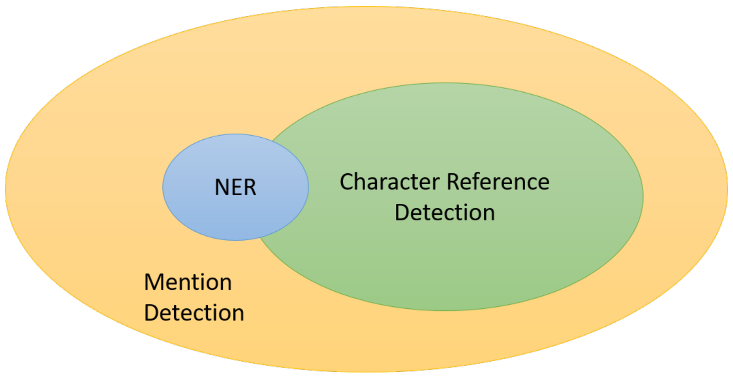


Figure 6.2.: All Character References can be considered as a subset of all noun phrases, which are in the center of Mention Detection. The character references that appear as common nouns overlap with the usual definition of a name in NER.

effect was even stronger, since the corpus consists solely of sampled sentences from German newspapers and Wikipedia articles. Since then, NER has become a task in which mostly local methods competed, even though global methods were developed as well (e.g. Skip-Chain Conditional Random Fields [Sutton and McCallum, 2012]). Nowadays, the classical NER scenario usually distinguishes four classes - Persons, Locations, Organizations and Miscellaneous, for names not fitting into any of the three prior classes. At the shared task at CoNLL, classifiers based on Maximum Entropy were the most dominant approach [Berger et al., 1996]. CoNLL 2003 was not only a turning point for rule-based systems, but also the emergence of Long-Short-Term-Memory (LSTM) Networks for NER. Interestingly, the system based on LSTMs [Hochreiter and Schmidhuber, 1997] came in last in the shared task.

With the extension of Maximum Entropy classifiers to handle sequential classification problems, first Maximum Entropy Markov Models (MEMMs) [McCallum et al., 2000] and then Conditional Random Fields (CRFs) [Lafferty et al., 2001] appeared, and estavblished as the state of the art system for many years, especially with the release of the tagger of Stanford [Finkel et al., 2005]. With regard to the features, CRFs made mainly use of word and character n-grams surrounding a focus token for which a label is to be predicted. Compiling gazetteers from unlabeled data (e.g. by using clusters from the Brown clustering [Brown et al., 1992b]), a technique called semantic generalization, usually improved the quality of existing systems [Faruqui et al., 2010]. The features derived from unlabeled data originated from different methods, such as LDA [Blei et al., 2003] or later on Word2Vec [Mikolov et al., 2013] [Jannidis et al., 2017]. Nowadays, features are no longer compiled by hand, but are derived by neural networks instead. Starting with different word embeddings (such as Word2Vec, GloVe [Pennington et al., 2014] or FastText [Joulin et al., 2016]), Bidirectional LSTMs encode the sentence and a CRF classifier at the top of the network predicts the output for an incoming sentence [Huang et al., 2015]. The work of [Riedl and Padó, 2018] used different available corpora for German NER and pretrained on the ones that were not used for evaluation and resulted in state of the art F1-score of 84.73% on the German CoNLL data and about 82.93% for GermEval. For english the incorporation of embeddings derived from bidirectional language models (such as ELMo [Peters et al., 2018] or BERT [Devlin et al., 2018b]) instead of just using pretrained word embeddings improved the state of the art for the English portion of the CoNLL data to an F1-score of about 92.22%.

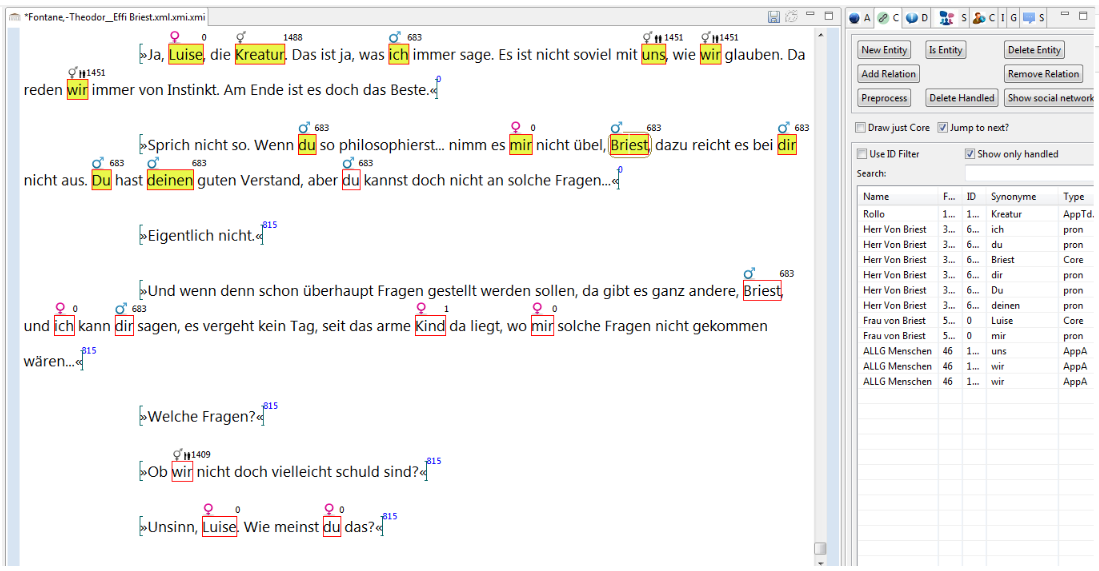
6.3. Data and Preprocessing

This section provides an overview of the necessary preprocessing that is required aside from the algorithms that perform the classification. In essence, this section describes the process of how our data was labeled and the resulting characteristics of that data. This work made use of DROC and the Kindler dataset, both are described more detail in chapter 4.

6.3.1. Annotation and Text Highlighting ENvironment (ATHEN)

This section gives a brief overview over the annotation tool ATHEN (presented at DHD 2018 [Krug et al., 2018b]) and its core ideas for annotating character references and coreferences during one pass over the text. Recapping DROC, we annotated mentions as a continuous span of text and a 'coreference link' as an attribute of such a mention (more precisely it is stored as an ID, unique for each entity in the text). In an efficient way to annotate this information, the reader should only pass a single time over the text they are currently working on. This was the core principle of designing the so called 'Coreference View', the main perspective in ATHEN, used to annotate documents with required coreferential information. It is depicted in figure 6.3.

The usual process of annotation is as follows: First the user can (if they want to) make use of the built in preprocessing tools (technically speaking, this is a pipeline built in Apache UIMA [Gotz and Suhre, 2004]) and obtain suggestions in the text, highlighted as red boxes around text. In early stages, this was a relatively simple script, written in Apache UIMA Ruta and is now replaced with a machine learning algorithm (described in section 6.7). Then the user reads through the text and either accepts or changes the proposed annotations. Changing the cluster ID of a reference is as simple as double clicking another reference in the editor (with the target ID) or double clicking a reference in the table on the right side of ATHEN. Following this procedure, the user only needs a single pass through the text while still being able to focus on the task. Once done, the user can now change morphological features of the entities in the table (that is selecting the correct gender or number, which can be done on entity level). Afterwards the user can save the document and continue with the next one. The automatic preprocessing is currently only capable of suggesting the spans and does not predict IDs. During our annotation phase, our annotators found the information from an automatic coreference system to be more confusing than actually helpful, while introducing a significantly longer waiting time for the preprocessing.



»Ja, Luise, die Kreatur. Das ist ja, was ich immer sage. Es ist nicht soviel mit uns, wie wir glauben. Da reden wir immer von Instinkt. Am Ende ist es doch das Beste.«

»Sprich nicht so. Wenn du so philosophierst... nimm es mir nicht übel, Briest, dazu reicht es bei dir nicht aus. Du hast deinen guten Verstand, aber du kannst doch nicht an solche Fragen...«

»Eigentlich nicht.«

»Und wenn denn schon überhaupt Fragen gestellt werden sollen, da gibt es ganz andere, Briest, und ich kann dir sagen, es vergeht kein Tag, seit das arme Kind da liegt, wo mir solche Fragen nicht gekommen wären...«

»Welche Fragen?«

»Ob wir nicht doch vielleicht schuld sind?«

»Unsinn, Luise. Wie meinst du das?«

☐ Draw just Core ☑ Jump to next?

☐ Use ID Filter ☑ Show only handled

Search:

Name	F...	ID	Synonyme	Type
Rollo	1...	1...	Kreatur	AppTd.
Herr Von Briest	3...	6...	ich	pron
Herr Von Briest	3...	6...	du	pron
Herr Von Briest	3...	6...	Briest	Core
Herr Von Briest	3...	6...	dir	pron
Herr Von Briest	3...	6...	Du	pron
Herr Von Briest	3...	6...	deinen	pron
Frau von Briest	5...	0	Luise	Core
Frau von Briest	5...	0	mir	pron
ALLG Menschen	46	1...	uns	AppA
ALLG Menschen	46	1...	wir	AppA
ALLG Menschen	46	1...	wir	AppA

Figure 6.3.: The coreference view of ATHEN: The left side shows the editor with the text that is currently worked on. Each text segment highlighted in yellow shows a character reference which was already accepted by the annotator, with the according entity ID written at the top right of the snippet. All references that have the same ID refer to same entity. The gender and number of a reference is depicted by appropriate symbols above the mention. A snippet which is just displayed by an unfilled box refers to an entity which has not been accepted yet and might originate from automatic preprocessing. The right side of the image shows the actual view, with all accepted references in a table which allows a fine grained editing of morphological information or the aggregation to entities which facilitates the annotation process.

6.3.2. Preprocessing of German Novels

This section recaps the tools that were used to preprocess the texts in the literary domain for the experiments in this chapter. Since there were no annotated corpora in this domain we were limited to existing tools for the preprocessing of the raw text sources. This section elaborates which tools were used.

Tokenization Tokenization of literary texts is not as demanding as in other domains such as medical texts, which contain a lot of abbreviations. The main challenges are the treatment of hyphenated tokens, the recognition of different quotation markers as well as the recognition of apostrophes *'s*. Even though literary texts may contain a substantial amount of dialects, they seldom introduce additional challenges to the tokenizer. Therefore the German model of the Apache OpenNLP tokenizer[3] yields good results in the literary domain

[3]http://opennlp.sourceforge.net/models-1.5/, accessed 20.05.2020

Sentence Splitting A task that is especially inter-woven with tokenization is the task of determining sentences. Since abbreviations are seldom (the ones we found are rather commons ones such as Dr.) and could be handled by the German model of Apache OpenNLP sentence splitter, we selected this component for sentence splitting.

Part of speech (POS) Tagging and Morphological Information The determination of POS tags for each token in a different domain is challenging. The usage of the RFTagger [Schmid and Laws, 2008] and the TreeTagger [Schmid, 1995] proved to be robust and fast and were therefore used as the main component to detect POS Tags, as well as gender, number and person information.

Lemmatization With its huge lexicon, the TreeTagger [Schmid, 1995] returns lemmas of high quality. On top, if a token is not known the user can decide to either get a default lemma or the token $< Unknown >$ which is helpful for downstream tasks such as CRD (since unknown words have a high chance to represent names, especially if they appear frequently).

Parsing The most challenging step in the preprocessing pipeline is syntactical parsing. If a flat hierarchy is sufficient for the task, then the chunker, which is integrated in the TreeTagger is used. If a full hierarchy is required, then the dependency parser of the MATE toolkit [Bohnet, 2010] has proven to yield both good quality and acceptable speed. Nevertheless the parser is still error prone in the literary domain, especially when applied in sentences that are located inside or nearby direct speech utterances, since those are barely - if at all - prevalent in German tree banks such as TIGER, TueBa-DZ [Telljohann et al., 2006] or NEGRA [Skut et al., 1997]. We found the parser to yield good quality in the determination of noun phrases and an acceptable quality for subjects, but a rather bad quality for genitives and datives. Even though PARZU proved to be even more robust, the experiments in this chapter have all been conducted using the results of the Mate Parser.

Word Category Determination Once all character references have been detected, this algorithm determines for each token of each reference a semantic category, from a list of available categories, depicted in figure 6.4.

In the first stage, the algorithm checks the gazetteers from section 6.4.1 and attaches the category of the list to the token. If the assignment is complete (e.g. there is no unknown labeling in any of the tokens remaining), then it is assumed to be correct and is not changed afterwards. For both other cases, the first one being the presence of unknown tokens, a list of templates (a template could be of the form "First Name + Last Name") is matched against the reference, each matching template is stored at the reference. All tokens that are labeled as unknown afterwards will not be revised and are left as unknown. The other case appears when multiple labelings of a token/multiple tokens are possible (Either due to the look up or the templates) and a strategy for resolution has to be applied. In those cases, a set of hand crafted preferences is applied

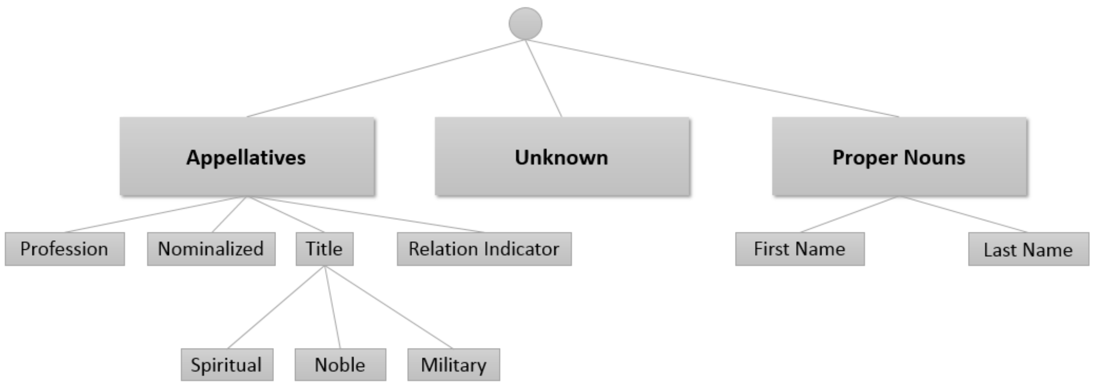


Figure 6.4.: The hierarchical representation of labels for each word. As input the algorithm gets to know whether a character reference is an appellative or a proper noun. The output is a label (a leaf node) from this hierarchy for each token.

that decides which templates should be kept in favor of others (e.g. if a token could be both - a name and title - then only the title is kept).

6.4. Adapting Existing Systems to the Literary Domain

Coreference Resolution for German newspaper articles or scientific articles has been researched extensively throughout the last decade. For its input, CR relies on so called mentions. In German literary texts, its usually the actions of literary characters that form the plot. This requires a character reference detection (CRD) done prior to (or in combination with, see [Lee et al., 2017]) the actual coreference resolution. In this work, we treated CRD with techniques of NER. This can be justified since its sole purpose is to detect spans of text which refer to literary characters. Span detection is a form of sequential classification and is the dominant technique of NER in the literature.

This section examines the tools and corpora that are already available for the German language and briefly describes the key aspects in which they differ from German novels - in terms of annotation guidelines and origin of the text material. Starting with NER, existing corpora resulted from Shared Tasks of different conferences. Most notably for this work, the corpus of CoNLL 2003 [Sang and De Meulder, 2003] as well as the corpus of GermEval 2014 [Benikova et al., 2014] are used alongside DROC.

The three corpora differ in multiple aspects: The most dominant criterion is the distinction by the types of references that are annotated. Both CoNLL and GermEval annotate four kinds of entities:

- PER - all person names
- LOC - all names of locations (such as rivers, cities, etc..)

- ORG - all names of organizations (such as Google, IBM, ...)
- MISC - all names that don't fit in any of the previous categories (e.g. Book titles or movie titles)

However for each of these categories, only those spans have been annotated which refer to a proper noun or to an entity which can be "uniquely identified" by this span (from now on referred to as *proper nouns*). If one focuses on a downstream CR between literary characters, only one of these categories is required ("PER") but in contrast to their guidelines all references to an entity are required and not just proper nouns. More specifically this includes common noun phrases and pronouns. While filtering the three additional categories is an easy step (e.g. by just discarding them before or after the training/prediction), creating an algorithm which predicts not only proper nouns but also common nouns and pronouns that refer to literary characters is challenging.

The first step is to identify the problem at hand: The required algorithm does not only need to yield good quality on a different domain (the transfer in this case is from news and Wikipedia articles to the literary domain) but also learn to predict pronouns and common nouns which are not even annotated in the source domain. The first problem is a classical instance of *domain adaptation* where the algorithm needs to generalize on data of one domain (*source domain*) which it is built upon and subsequently gets applied to text of the *target domain.* In this particular instance the main challenges are to overcome the gap in vocabulary as well as the differing grammar when shifting from newspaper texts to literary novels.

The second problem is to introduce new "categories" into the prediction (namely common noun phrases and pronouns). This is generally referred to as *transfer learning* or multi-task learning (in this setting, the prediction of an additional category can be viewed as a separate task which is performed alongside of the original one). Section 6.4.1 provides the details for the adaptation experiments in this work.

6.4.1. Transfer Learning from NER to Literary Character Reference Detection

At the beginning of our project, no annotated data in the target domain was available. We therefore experimented with techniques from semi-supervised learning in order to integrate common noun phrases and pronouns which are required for CRD into a classifier. Specifically, we had the training and development data of CoNLL and GermEval 2014 at hand. As DROC is now available, it can be used to evaluate the different approaches for the adaptation of NER to literary CRD. Additionally, we had access to about 450 novels from project TextGrid and about 1500 from project Gutenberg[4], both corpora do not contain any manual annotations. We performed three experiments to automatically enhance the capabilities of a classifier trained on existing corpora.

The first and most uninvolved experiment is to just train the proper nouns on the existing data and add pronouns and common nouns (appellatives) using gazetteers. This

[4]`https://www.gutenberg.org/`, accessed 20.05.2020

approach serves as a baseline for more involved approaches. The second experiment is situated in the domain of *distant supervision.* We train a classifier for proper nouns on the news corpora and apply them to the full text of 90 novels of project TextGrid (the same 90 novels, which are used to extract the segments for DROC). For appellatives and pronouns, we used gazetteers and labeled these novels. These *weak labels* are then used to train the classifier. The third approach originated in semi-supervised learning. Instead of training the classifier by enriching the instances with features, we directly provide the importance of a feature using expectations and train a Maximum Entropy classifier using Generalized Expectations [Druck et al., 2008].

All experiments make use of the entire DROC-corpus as evaluation data, that is about 400.000 tokens of evaluation material. The metrics of choice are labelled and unlabelled entity-F1, a true positive (TP) can therefore only be obtained by correctly predicting the entire span of a reference, as well as the correct label in the labelled setting. To highlight the situation at the beginning of the project, we used the (first order, linear chain) Conditional Random Field classifier of Stanford [Finkel et al., 2005], with semantic generalization features integrated in the model released by Pado [Faruqui et al., 2010] using the Dewac corpus[5] as our base classifier for the detection of the proper nouns.

Gazetteers for Appellatives and Pronouns

This section addresses the problem of acquiring gazetteers for CRD. The experimental scenarios described in 6.4.1 require gazetteers for appellatives and pronouns. Getting a list of pronouns is fairly easy (e.g. by compiling a list out of the lexicon of the RFTagger since pronouns form a closed class of part of speech tags (aside from different dialects)). Finding a suitable list of appellatives is more challenging. We compiled lists of military titles, spiritual titles, professions and noble titles from Wikipedia[6] and used the semantic field "Mensch" ("human") of GermaNet [Hamp and Feldweg, 1997] (the German counterpart to WordNet [Miller, 1998]) and compiled all words into separate lists. This results in five different lists, one for each category. Furthermore we extracted the following lists (these are all based on the previous lists) from the tokens of our 450 novels, derived from project TextGrid.

1. GermaNetLemma: If a lemma of a token in the novels is mapped to "Mensch" in GermaNet

2. GermaNetStem: If a stem of a token in the novels is mapped to "Mensch" in GermaNet

3. GermaNetSuffix: If a token in the novels ends on an entry (it is also enforced that the remaining prefix is a valid word according to a lexicon) with category "Mensch" in GermaNet (e.g. "Artilleriefahnenjunker" ends on "Junker" which is of category "Mensch" and Artilleriefahnen is still a valid German word)

[5]`https://www.sketchengine.eu/dewac-german-corpus/`, accessed 20.05.2020
[6]E.g. for noble titles, see: `https://de.wikipedia.org/wiki/Adelstitel`, accessed 20.05.2020

4. GermaNetW2V: If the Word2vec-Embedding[7] of a token was clustered (as derived by k-Means [Lloyd, 1982]) in a cluster with at least 80% tokens which can be mapped to GermaNet and show the category ”Mensch”

In the same manner, additional 16 lists (four for each category) were generated by using our previously extracted titles and professions. In German, every verb as well as every adjective can be converted into a noun (e.g. ”gehen” → ”Gehende” or ”lebhaft” → ”Lebhafte”) by using the base of the verb/adjective and appending suffixes according to female or male morphology (e.g. ”-er, -en, -em, -e, es”). Additional two lists are compiled by converting every verb/adjective of the novel corpus into a noun. In total, this resulted in 27 lists which were used to determine appellatives which refer to literary characters. This process takes mainly care of different morphology (lemma, stems), compound words, which are particularly frequent in German texts and generalizes the lists by making use of Word2Vec Embeddings, previously calculated using the 1500 novels from project Gutenberg and 100 Dimensions using default parameters of the skip gram architecture.

6.4.2. Results for the Different Methods of Domain Adaptation

This section contains the results of the aforementioned experiments for the domain adaptation. For all experiments the entire corpus DROC is used for evaluation. As baseline, the Stanford CRF with the previously mentioned model is applied and evaluated for different scenarios, the results are summarized in table 6.1.

Table 6.1.: Results of the Stanford NER model on the DROC corpus. If all character references of DROC are included in the evaluation, the classifier only reaches an F1-score of about 8%. If only proper nouns are taken into account, the entity-F1 rises to 41%.

Scenario	**Precision**	**Recall**	**F1**
Stanford NER all References	42.15	4.62	8.33
Stanford NER proper nouns	42.15	40.24	41.17

The baseline obtains a rather low Precision of about 42%, but a really low Recall with just about 5% score. This is because the classifier was not trained for the detection of neither common nouns nor pronouns and the majority of all character references consist of pronouns.

The first scenario for the adaptation, as shown in table 6.2 is the combination of the extracted lists with the baseline classifier. The generated gazetteers yield an F1-score of about 58%, when evaluated against the appellatives in DROC, with the score heavily shifted towards the Recall. In combination with the baseline classifier, this raises the classification F1-score to about 27%, increasing the Recall by about 15%. Using a list with all pronouns from a dictionary, and removing all neuter pronouns and all pronouns

[7]We used the standard implementation with default parameters in the skip gram setting on 1500 novels from project Gutenberg

which could also represent articles (yielding a total of 245 pronouns) reaches a score of 81% when evaluated against the pronouns of DROC. Unsurprisingly the usage of every instance of the pronouns of the list yields a Precision of just about 72%, but a very high Recall of more than 92%. The combination of the baseline classifier with the gazetteers for appellatives and pronouns produces an F1-score of 72%. However, since about 2/3 of all character references are pronouns, this score is dominated by the quality of the pronouns itself.

Table 6.2.: Integration of the extracted gazetteers into the baseline classifier yields a maximum F1-score of about 72%, with a good Recall but a mediocre Precision of just about 63%

Scenario	**Precision**	**Recall**	**F1**
Gazetteers on Appellatives	47.97	75.14	58.56
Baseline +Gazetteers	46.44	19.35	27.31
Pronoun list on pronouns	71.55	92.61	80.73
Baseline +Gazetteers + pronoun list	63.56	83.13	**72.04**

The second experiment features ideas based on distant supervision, the baseline system as well as the tokens in the gazetteers are annotated in the 90 full novels prevalent in DROC. The Stanford CRF is then retrained using this data set and evaluated on DROC. We used the same 90 novels for training to give an optimistic estimation of what score could be achieved by applying such a technique. In total this created about 8.7 million tokens for the training of the CRF. The results of this approach can be seen in table 6.3.

Table 6.3.: The results of the Stanford CRF retrained on 90 fully automatically labeled novels. Making use of weak labels improved F1-scores on all three classes.

Scenario	**Precision**	**Recall**	**F1**
Unlabeled Score	72.45	84.30	**77.93**
Labeled Score	71.90	83.66	77.34
Score on proper nouns	64.89	39.84	49.37
Score on appellatives	59.31	73.54	65.66
Score on pronouns	76.08	93.87	84.05

The training on the automatically labeled data in the target domain yields an improvement in all three categories compared to the baseline experiment. The F1-score for proper nouns raises from about 41% to almost 50%, and shows the largest improvement in terms of absolute numbers. But the other two categories could also be predicted more accurately afterwards. The F1-score for appellatives raised from 58.6% to 65.7% and even pronouns reach a score of about 84%. In total the labeled F1-score raised by almost 6% from 72% to almost 77.5%. Since no additional preprocessing was made on the texts as just the tokens have been fed as training material into the CRF, the sheer inconsistencies of the context features must have been the cause for this improvement.

We note, that those numbers are rather optimistic, since we specifically chose the same novels for the semi-supervised training that were afterwards used for evaluation.

The next approach makes use of a prominent technique of transfer learning. We use both data sets, CoNLL and GermEval, and train a Maximum Entropy classifier on the combination of both data sets. In order to also integrate the detection of appellatives and pronouns, we integrate the knowledge gathered by the gazetteers into the training step by converting this knowledge into so called Generalized Expectations as follows:

1. The gazetteers and pronouns are annotated onto both data sets
2. The feature set of table 6.8 is extracted from those two documents.
3. For each extracted feature, the label distribution is derived by counting the weakly labeled data.
4. If the label distribution for a feature shows 100% for any label other than O (which means that the token is not part of an entity), the probability of O is set to 0.05 and the probability of the other label is set to 0.95. (This is to model the uncertainty introduced by the automatically extracted gazetteers)
5. Each of the resulting label distributions are then converted into a constraint.

The annotation to the data keeps the annotations that are labeled as *PER*, but we don't just simply discard the information that something is a named entity of a different category. Since the gazetteers are extracted in a fully automatic process, they are noisy and contain wrong entries. We therefore did not annotate locations or organizations as appellatives, even though they are part of the gazetteers. This ensures that we could incorporate the information of the other labels as well.

Generalized Expectations modify the objective function $\mathcal{O}$, by adding another regularization term:

$$\mathcal{O}(x, y, \theta) = \mathcal{L}(x, y_{labeled}, \theta) + GE(x, y_{unlabeled}, \theta) + Reg(\theta)$$

With (x, y) being a single instance for the classifier, θ the parameters of the model, $\mathcal{L}$ being the likelihood on the labeled data set, GE being the expectations that are posed towards the classifier on the unlabeled data set and Reg is the standard regularization term such as a L_2 or L_1. The labels for $y_{labeled}$ originate from the labeled data sets and the labels $y_{unlabeled}$ are converted into the form of label distributions (the expectations) of individual features. For this experiment, the Expectation constraints try to minimize the Kullback-Leibler divergence of the provided feature label distributions and the ones predicted by the model. The results of this experiment are depicted in table 6.4

This experiment outperforms the other two approaches, reaching a labeled F1-score of about 82%, while improving on the pronouns and proper nouns categories. However the scores of especially the appellatives are still rather low with just a mere 60% Precision and Recall. Apparently even though we integrated a small margin of uncertainty to the

Table 6.4.: The results of the Maximum Entropy classifier trained using Generalized Expectations and evaluated on the full DROC data set. With an F1-score of 82.6% this experiments shows the highest results in our task given just gazetteers and data from a different domain and annotated with different guidelines.

Scenario	**Precision**	**Recall**	**F1**
Unlabeled Score	78.93	86.61	82.60
Labeled Score	76.21	83.62	79.74
Score on proper nouns	78.28	62.32	69.40
Score on appellatives	62.66	60.44	61.52
Score on pronouns	79.80	95.19	86.82

labeling of the data, the classifier fails to completely make use of that information and to improve on the appellatives. Even though the entire F1-score improved by more than 10%, this is mainly due to proper nouns and pronouns. In order to improve the quality of the appellatives, it appears that different strategies have to be employed. The next section tries to overcome this using Active Learning.

6.5. Active Learning for Literary Character Reference Detection

If the goal is just to get a CRD system with reasonable quality, a different strategy can be followed (this has to be mentioned, since Active Learning as it is used in this work does not produce entire labeled documents and thus this data can not be used for subsequent coreference resolution). First, one starts with a small amount of training data, trains a classifier (or an ensemble) and applies the classifier to a set of unlabeled documents. Then, within those documents the next most valuable (according to a given query strategy) instances are selected to be labeled by an annotator. This process is referred to as *Active Learning*. In this section we show that if the sole goal is to get a CRD of good quality it is sufficient to label only a small fraction of data. In our setting we start with a seed of 50 labeled instances and continuously add 250 more instances until a maximum of 50.000 instances of the training data have been included in the training process. An instance in this setting is a single token with its label. We compare the following query strategies (a good overview can be found in [Settles, 2009]) for the selection of the most informative instances:

Query By Uncertainty The classifier is trained on all previously selected instances and applied to all remaining instances. The instances are selected by the minimum classification probability $x_{UC} = argmin_x P(y|x)$

Table 6.5.: Results of the classifier for the different query strategies, in all settings, 10050 training instances were presented to the classifier.

Classifier	Query-Strategy	$F1_{ul}$	$F1_{core}$	$F1_{app}$	$F1_{pron}$
MaxEnt	Entropy	91.6	79.0	80.8	93.2
Perceptron	Entropy	86.2	50.5	69.1	92.1
MaxEnt	Margin	**92.0**	**80.5**	**81.5**	**93.4**
Perceptron	Margin	87.5	64.1	75.4	91.7
MaxEnt	Uncertainty	92.0	80.0	81.3	93.3
Perceptron	Uncertainty	86.9	61.6	74.7	91.4
MaxEnt	Naive	87.1	59.0	69.6	92.2
Perceptron	Naive	86.2	50.0	69.4	92.0

Margin The classifier is trained on all previously selected instances and applied to all remaining instances. The instances are selected by the minimum classification margin between the highest two classes $x_M = argmin_x[P(y_1|x) - P(y_2|x)]$, with y_1 being the highest scored label and y_2 being the label with second highest probability.

Entropy The classifier is trained on all previously selected instances and applied to all remaining instances. The instances are selected by maximum entropy of the resulting probability distributions, calculated on all instances: $x_{En} = argmax_x H(y|x)$.

Naive The classifier is trained on all previously selected instances. The instances that are chosen are just the first instances of the unlabeled instances.

The feature set which was used for those experiments is depicted in table 6.8. We compared two different classifiers, namely a Maximum Entropy classifier using an L2-Loss with a prior standard deviation of 1.0 and an Averaged Perceptron algorithm [Collins, 2002] using a margin-objective of 1 (see [Crammer et al., 2006]). The Maximum Entropy classifier is trained until the log-likelihood improvement between epochs fell below a tolerance of 0.0001 and the Perceptron is trained until no parameter updates could be made or up to a maximum of 50 epochs. The results are shown in figure 6.5.

The results of CRD with Active Learning show, that in order to obtain a high quality component it is sufficient to label about 10.000 instances (see table 6.5 for exact numbers). In both cases, the Query by Uncertainty and the Margin objective, a peak performance of about 92% unlabeled F1 could be obtained. The naive setting reaches about 87% F1 with that amount of labeled data. However after these 10.000 instances, the quality of these approaches grows very slowly, in fact compared to the results with the supervised setting that has access to the entire training set (see section 6.7), there is a mere improvement of about 0.2% F1 score. Since the features of this approach are mainly local to the token, after about 10.000 instances it appears to have reached a state in which new contexts appear sparse and don't help to generalize further. By integrating global consistencies into the classifier (see section 6.8), the classification accuracy can be further increased.

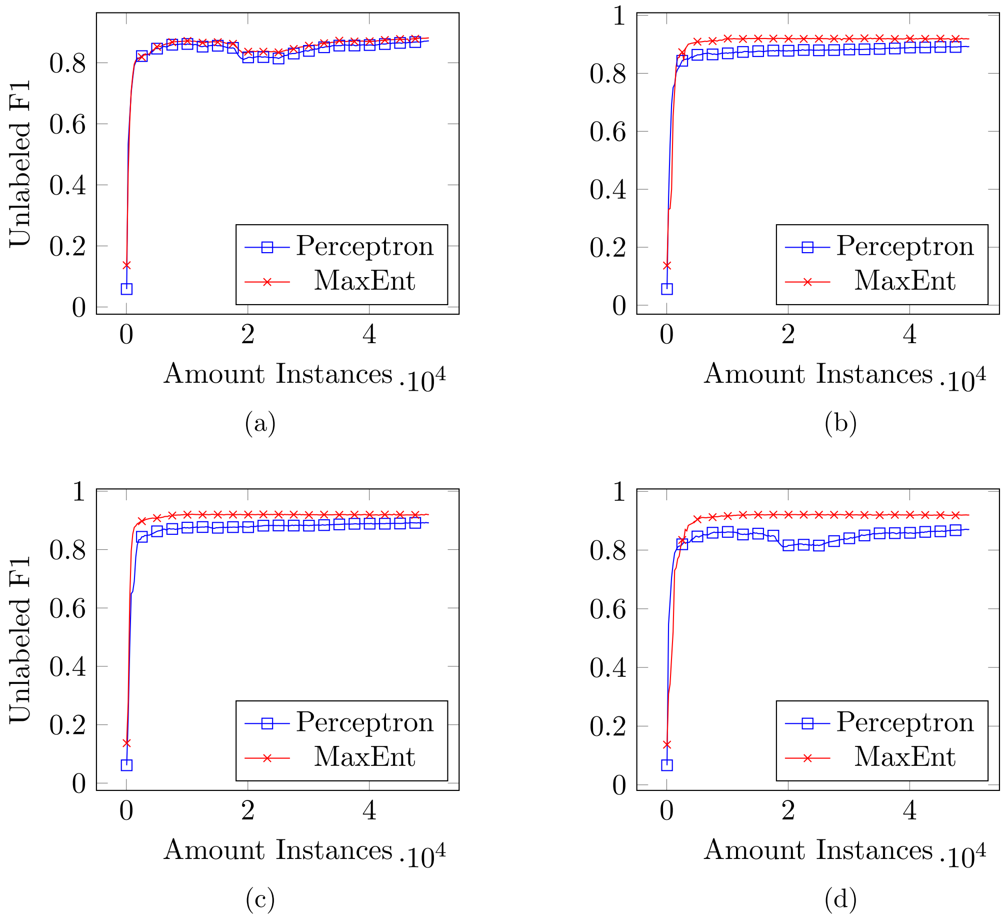


Figure 6.5.: Experiments using Active Learning, a) shows the naive strategy which only selects the next 250 instances, b) uses Query by Uncertainty, c) uses the margin objective and d) uses the Query by Entropy schema. In every scenario the MaxEnt classifier outperforms the Perceptron. In general, the quality rises very quick (This is mainly due to pronouns being selected first). The MaxEnt classifier reaches its peak performance of about 92% in the Margin setting and the Uncertainty setting with about 10.000 instances. The Query by Entropy setting requires about 15.000 but reaches the same peak performance. The quality in the naive setting reaches a maximum of 88% at 50.000 instances.

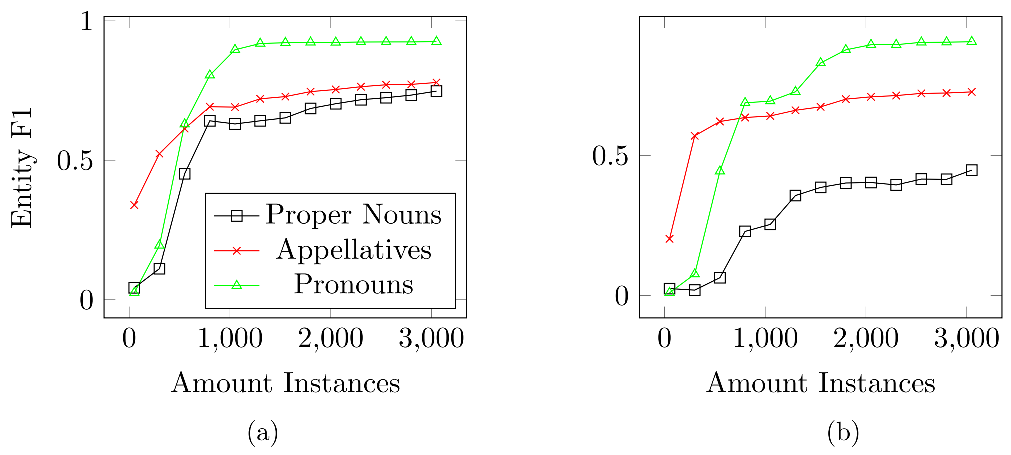


Figure 6.6.: Comparison of F1-scores for the different categories (Proper nouns, appellatives and pronouns) for the MaxEnt (a) and the Perceptron (b) using the Margin objective. In both cases, the quality of pronouns reaches its peak rather quick while the other categories ramp up to a mediocre value and then start to grow slowly.

One interesting aspect is to compare the classification accuracy throughout the three categories (proper nouns, appellatives and pronouns) during the Active Learning process. Figure 6.6 shows this characteristics for the Margin objective, for both the MaxEnt as well as the Perceptron. In both cases, the quality for pronouns reaches its peak rather quickly (which explains the steep ascent in the Active Learning curves), while the other categories grow much slower.

6.6. Creation of a Rule-Based System for Character Reference Detection

A different method of obtaining a character reference detection is by formulating domain knowledge directly in the form of rules. This has the advantage that no training data has to be available (it facilitates the process of creation, however). In academic publications it appears that rule-based systems are considered out of fashion. This might be due to a) they are cumbersome to create and maintain and b) the quality is considered to be lower than of machine learning approaches. While it is true that, if a task requires a combination of thousands of weak features, rule-based methods are very unlikely to yield comparable qualities. For the task at hand, though, the application can make use of substantial heuristics that are very hard to incorporate into a machine learning setting. The domain of this work is literary fiction, and within these texts characters usually are introduced and reappear many times over the course of the story. This can be exploited in a rule-based algorithm, since detecting a character means only detecting it once instead of repeatedly in every single sentence (which is the dominant approach for

NER with machine learning methods). We refer to a detection in individual sentences as a local detection and the propagation of this information as a global detection. Our algorithm makes heavy uses of those two different views of the text. In its core the algorithm is a scoring procedure - similar to most machine learning algorithms - with a severe distinction: All scores for a rule have been set manually by a rule engineer. In its current state, the algorithm predicts a scoring for each token of the input text, whether the token represents the beginning of a reference ("B"), is located inside a reference ("I") or is not part of a reference ("O"). In the last stage, the labels which received the highest scores are selected and the spans of a reference are recreated and labeled as name ("core"), appellative ("app") or pronoun ("pron").

The algorithm works as follows: First for every token a wrapper object is created, which is used to store all temporary information as well as all rules that have matched. The algorithm itself can be separated into five steps, each operating on the results of the previous steps. The first four of these steps will only assign positive scores to a token (which means it receives a "B" label at that time) or negative scores (referred to as downvoting) which means it gets an "O" assigned. Only the last step will take care of building the correct spans for the references and assign "I" labels.

Step 1: Precise Constructs (Local) This step analyzes the token's text, its POS-tag and its neighborhood and assigns positive or negative scores. A few examples:

- All POS-Tags besides nouns and pronouns get downvoted, all pronouns upvoted (a little)
- "Gott" (God) in special context gets downvoted
- Temporal expressions, locations, organizations, monetary units get downvoted
- When dialects are used (e.g. Bavarian and Berlin German), especially the pronouns get upvoted
- Exclamations (such as *"Aha"* or *"Oho"*) get downvoted
- Special constructions based on covering noun phrases get up- (e.g. *"Seine Majestät"*) or downvoted
- A token in subject position with a head (according to the Mate Dependency Parser) which can be found in a list of communication verbs gets upvoted. (*speaker*)
- Tokens in direct speech which are surrounded by tokens of length 1 get upvoted. (*addressee*)

Step 2: Gazetteers (Local) All gazetteers from section 6.4.1 as well as additional lists of male and female first names are matched onto the tokens, their lemma and stems. In the same manner, GermaNet is queried, and the resulting semantic fields[8] are analyzed.

[8]Semantic fields are depicted here: `http://www.sfs.uni-tuebingen.de/GermaNet/germanet_structure.shtml`, accessed 20.05.2020

If the token is known by GermaNet but no character related category is assigned to it, it gets downvoted. If the category is "Mensch" (human) only, then it gets upvoted. If more than one category is determined, then we employ a simple disambiguation schema.

Furthermore this stage splits all remaining, yet unknown tokens, if they start uppercase. The splitting procedure maps all sub strings onto a dictionary[9], and whenever the entire text can be split into entries from the dictionary (including so called "Fugenelemente" *(According joints)*), then a valid split is returned. If valid splits exist, we map the last entry of the splits onto GermaNet and utilize the same mapping schema as described above to assign scores to the token.

Step 3: Propagation (Global) This step is the first one which can be classified as a global step, since it makes use of the previously assigned scores and the resulting best labels for each token. All remaining tokens that are yet unassigned are downvoted, if they start lowercase. In the same manner, if unassigned tokens show a special suffix, then an according score is assigned.

The next procedures work on a global scale. We strip the inflection of all tokens and group all tokens by the remaining body. These groups are then evaluated and the most certain label that originates from the sum of all entries in that group is spread to the other members of the same group.

In the next step we build so called *CW-neighborhoods*, (Capital Word neighborhood) from the tokens, that is we expand a token to the left and to the right as far as capital words can be detected. Then for a string of a token, all CW-neighbourhoods are compared to a list of valid templates (e.g. a template is of the form *profession→first name →UNKNOWN* in which case the Unknown token will be upvoted as well.). We utilize a similar procedure if tokens adjacent to a previously detected reference are labeled as a name by the POS-Tagger.

We furthermore constrain tokens in enumerations and appositions to the same label. We extract all head nouns of those constructions using the dependency tree and spread the most certain label to the other entries to ensure consistency.

All remaining unknown tokens are counted, and if they appear more than 3 times and are not part of one of the previously extracted locations (they were extracted during the first step), then they get upvoted as well.

Step 4: Cleaning This step reconsiders tokens that have been assigned a "B" label until now. If this token is part of the location lists (extracted in step 1) and has no adjacent tokens which start uppercase, then they get downvoted. The other step is attributed to the labeling schema of DROC. Tokens that appear inside negated scopes were not annotated (e.g. "kein Mann" (no man)) and are therefore downvoted again.

Step 5: Expansion This last step uses all tokens with their prior information whether they appear to be part of a reference "B" or not. All adjacent tokens with a "B" are collected and their attached information is matched against special *construction templates*

[9]The dictionary we use is included in the model of the RFTagger

Table 6.6.: Results of the rule-based algorithm. It is noteworthy that the results on the test set are higher than on the training data.

Experiment	$F1_{ul}$	$F1_{lab}$	$F1_{core}$	$F1_{app}$	$F1_{pron}$
Rules on test	91.30	86.93	72.13	75.91	92.61
Rules on train	90.75	85.49	77.10	73.67	90.28

(one construction template is of the form Title→Male name⇒ B→I, for example). In total we employ about 100 of these templates and an additional 25 that are dedicated to handle genitive constructions.

One thing to note in this pass is the way it handles constructions, such as the following (artificial) example

```
Da betrachtet Harry Ron vorwurfsvoll
(Harry looks at Ron reproachfully)
```

In this case, Harry and Ron are different entities but happen to be direct neighboring tokens. In those situations usually both - our chunker and our dependency parser- fail to detect this. However merging those tokens into one entity introduces severe problems for coreference resolution, since it creates the risk of merging two main characters into a single entity. We therefore inspect 100 tokens prior and after an instance and search for the individual strings (for the example we would search for Harry and Ron, whether they appear individually as well). If they do, then we do not merge them into the same reference. The category ("core", "appellative" or "pronoun") is determined using the attached list and template information. Afterwards all phrases can be built and the algorithm returns.

The algorithm is evaluated on the test split and on the training documents separately. None of the test documents have been inspected during the creation of the rules. The results are depicted in table 6.6.

The rule-based algorithm reached an unlabeled score of 91.3% on the test set and 90.8% on the training data. Interestingly, even though the training data was inspected during the creation of the rule-based algorithm (especially for the values of the scores that are assigned to each rule). This shows that, at least for this domain, the results appear to be rather stable. One major weakness of the algorithm is that its score drops to about 87% when its asked to also predict the label. In general the algorithm predicts a core entity, whenever a token is unknown, which yields to an over prediction of core entities (They only show a Precision of about 63% on the test data, while the Recall of the appellatives shows about 64%). Evaluating whether the rule based algorithm can differ between a pronoun and a noun reveals an F1 of about 81.5%. This means that the vast difference between the unlabeled and the labeled score is caused by a misclassification between appellatives and proper nouns. Another rather untypical property of the rule-based system is the ratio between Precision and Recall. In the unlabeled setting, the Recall was 88% and the Precision about 86%, while usually rule-based systems show much higher Precision and lower Recall.

6.7. Supervised Learning for Character Reference Detection

Traditionally, NER is solved using classical supervised learning. We conducted experiments for the prediction of character references using the 80% vs. 20% train/test split described in section 6.3 (54 training documents + 14 documents of the development set form the training data set) . In this section we compare three different approaches for the recognition of character references:

Unstructured Classification In this setting, every token of the training data is fed as input into the classifier and a classification label is predicted (”B-pron”, ”B-app”, ”B-core”, ”I” or ”O”). We experimented with the Maximum Entropy classifier and the Perceptron algorithm as classical approaches (using the feature set depicted in table 6.8, as well as Feed Forward Neural networks.

Structured Classification In this scenario, a sequence of tokens (in this case a sentence) gets predicted in one query of the classifier and a sequence of labels is returned. We experimented with Linear Chain Conditional Random Fields as well as BiLSTM-CRF neural networks. Both the traditional CRF as well as the Neural CRF predict the sentence using first order templates.

Ranking On top of the previous approaches we cast the prediction of character references as a ranking problem. The input to this system is a span of tokens with all possible combinations of labels. The ranker is then asked to score the label combinations with the input sequence. The predicted sequence is the one with the highest assigned score. Since for German texts one of the main challenges is to decide whether a capital word belongs to a character reference, the input instances are sequences of adjacent capital words. Tokens in lowercase are treated separately and the ranker just needs to decide whether this token forms a reference or not. An example of this approach is depicted in figure 6.7.

We employed the Perceptron algorithm using the Mira objective for its parameter updates [Crammer and Singer, 2003] in order to score the different combinations (referred to as Rankceptron). The features for ranking are depicted in table 6.7 Its neural counterpart is the mention detection stage of the end to end coreference system released by Lee et al. [Lee et al., 2017]. We retrained this model on our training set and applied it to the test data. Both ranking approaches are only able to predict the spans and can therefore only be compared on the unlabeled setting.

The results of this section are summarized in table 6.7. The results are evaluated using both, labeled and unlabeled entity-F1, as well as the F1-scores on the individual classes *core, appellative and pron.* The Perceptron was trained for 500 epochs using the margin objective and for 50 epochs using the Mira objective. MaxEnt was trained until the change in the likelihood fell below a tolerance of 0.0001 using the L-BFGS solver [Nocedal, 1980]. The Deep Learning approaches were trained for up to 100 epochs and

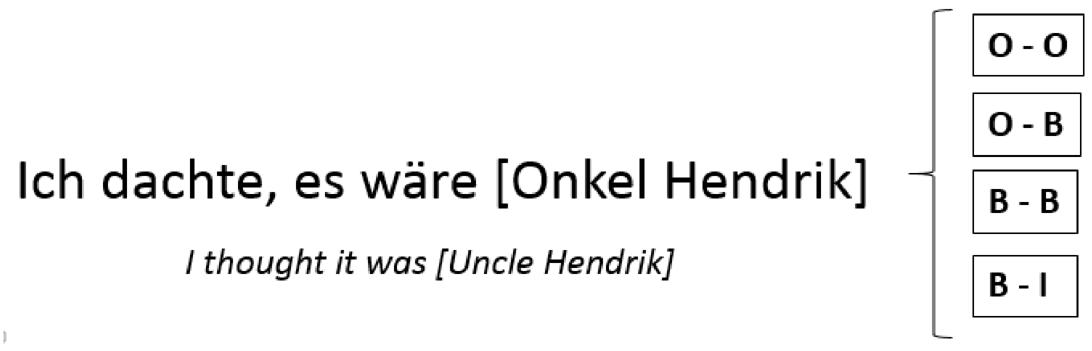

Ich dachte, es wäre [Onkel Hendrik]
I thought it was [Uncle Hendrik]
O - O
O - B
B - B
B - I

Figure 6.7.: The four different instances that are created for "Onkel Hendrik", a sequence of adjacent capital words. The ranker then has to score all four combinations and the label combination which gets ranked first is selected. Lowercase tokens as well as singleton capital words get assigned just two possibilities.

Table 6.7.: Results for the supervised experiments. All results are given in %, rounded to two digits

Experiment	$F1_{ul}$	$F1_{lab}$	$F1_{core}$	$F1_{app}$	$F1_{pron}$
Perc$_{margin}$	90.76	88.71	79.0	80.55	92.55
Perc$_{Mira}$	91.89	89.42	**81.35**	**81.40**	93.05
MaxEnt	**91.90**	**89.50**	80.73	81.32	93.26
FF	89.38	86.16	64.14	76.0	92.64
CRF	90.74	88.59	78.81	78.5	92.90
Rankceptron$_{Mira}$	88.74	-	-	-	-
E2E-Mention Detection	90.93	-	-	-	-
BiLSTM-CRF	91.85	89.47	79.48	79.36	**93.86**

the best model was chosen based on the quality of the test data[10]. The Feed Forward architecture uses a context of two tokens to the left and right and predicts a label for the token in the center. The tokens were fed using pretrained word2vec embeddings derived from 1.500 novels. It was trained using stochastic gradient descent with a learning rate of 0.05 and a batch size of 64. The network had three hidden layers of sizes 200, 150 and 100, this network architecture proved to be the most reliable Feed Forward architecture during prior experiments. The BiLSTM-CRF embedded the state to $2\times$ 64 hidden units and uses the same hyper parameters as the Feed Forward architecture.

The evaluation shows, that sequential methods only seem to provide a beneficial effect on the classification accuracy for the pronouns. This is not surprising since most tokens in the text are not part of a character reference and references tend to be rather short. The classical approaches, especially the MaxEnt classifier and the Perceptron using the Mira update show high classification scores, MaxEnt even outperforming BiLSTM-CRFs

[10]This gives an unfair advantage to the Deep Learning methods, however since otherwise getting in depth results would require a huge parameter study

Table 6.8.: Features used for the classical supervised machine learning approaches as well as for the Active Learning approaches.

Feature name	**Description**
Word, Lemma, POS	The token text, POS-Tag, Lemma of the token
Morphology	The gender, number and case as derived by the RFTagger
Gazetteers	Whether the token is in an extracted list
GermaNet-Category	The semantic field category in GermaNet or null
Word2Vec-Cluster	the clusterID of a previously created k-Means clustering of Word2Vec embeddings
Shape	Whether the token is Uppercase, as well as all suffixes up to a length of 6
Context	All previous features in the local context of the token in the intervals [-2,2], [-1,1], [-2,0], [-1,0], [0,1], [0,2]

on the labeled and unlabeled setting. Mira reaches highest F1-scores on appellatives and proper nouns and the BiLSTM-CRF the highest scoring on the pronouns. Since there is no dedicated feature integrated into the classical approaches which would help to distinguish between pronouns that refer to characters and those that do not, the ability to derive those *long range* features in LSTM layers is beneficial. Both ranking systems show lower scores, with the neural ranking model outperforming the Perceptron by more than 2%.

6.8. Combining Classifier Results

In order to further improve the results of the prediction of character references, we experimented with the combination of different approaches. More precisely we combined the classical MaxEnt classifier and the Perceptron using the Mira update with our rule-based approach. The easiest way of doing so, is to apply the rule-based system first and instead of using the scores that were selected by the rule engineer, the matching rules are converted into features and the scores are learned. A second option is to add these rule-based features to the regular feature set, shown in table 6.8. And a third way is to use the reliable heuristics of the rule-based approach in order to filter instances prior to the training, so that the classifier can focus on just the harder cases. The results of these experiments are summarized in table 6.9.

The scores (assigned to each rule) that were given manually yield a higher evaluation quality than the ones that were learned. It is noteworthy that these two experiments cannot be compared directly. Since the rules are applied one after another, the rule-

Table 6.9.: Results for the combination of rules with supervised learning methods. Best results for a category are marked in bold. Both, the integration of the rules as features, as well as using the rule-based algorithm to filter instances yield positive results for the classification accuracy.

Experiment	$F1_{ul}$	$F1_{lab}$	$F1_{core}$	$F1_{app}$	$F1_{pron}$
MaxEnt Just Rules	90.54	86.03	79.58	75.08	90.23
Perc$_{Mira}$ + *Rules*	92.71	89.93	**85.19**	81.70	93.26
MaxEnt + Rules	**92.88**	**90.21**	85.01	**82.05**	**93.52**
MaxEnt$_{Filtered}$	92.19	89.79	81.03	81.81	93.51
MaxEnt$_{Filtered}$+Rules	92.14	89.77	80.98	81.84	93.48

based system can make use of this order, while for the MaxEnt, just the results of all rules are attached as features. The integration of the rules into the standard set of features yields an improvement of almost 1% in the unlabeled and about 0.7% in the labeled setting. Since the rule algorithm itself can not distinguish between personal pronouns that refer to a character and those that do not, the improvement for pronouns are smallest. A huge improvement can be seen in the *core* category with an increase of almost 5%. These characteristics can be found in both algorithms, the MaxEnt as well as the Perceptron. An explanation for this is that by integrating rules as features, characters that appear many times can be tracked, even though the algorithm itself only captures features in a window of five tokens.
Filtering instances prior to training does yield a smaller gain compared to the integration of the rules as additional features.

6.9. Error Analysis

In order to make further improvements, there is a necessity to understand the remaining errors of the classifier. For this purpose, we chose four of our systems and manually analyzed their predictions on five documents of the test data. We chose to analyze the best ranking system (E2E-Mention Detection), the best combination with our rule set (MaxEnt + Rules), the BiLSTM-CRF and our rule-based system. For this purpose we annotated the results of those systems into the documents and compare the annotations using ATHEN. We chose the five documents by analyzing the results of the different systems on those documents, the overview is given in table 6.10.

The following section goes over the patterns that could be observed when manually comparing the gold entities with the ones predicted by the different algorithms. In many occasions it is very hard to exactly find the cause, this is why we restrict this analysis to only the most obvious weaknesses of the algorithms.

E2E In *Der Lehnhold*, there are passages that deal with the description of the environment or animals, and the algorithm mistreats the pronominal references as character references. Many of the false negatives are due to forgotten appellatives, especially since

Table 6.10.: Unlabeled F_1 scores in % on the 18 documents of the test set of the different approaches. Documents that are selected for error analysis are marked in bold.

Document\Approach	E2E	MaxEnt+R	BiLSTM	Rules
Ahlefeld- Charlotte von Erna	92.20	**93.52**	91.80	91.74
Anonym- Schwester Monika	93.03	**95.12**	92.71	94.06
Arnim- Goethes Briefwechsel mit einem Kinde	93.72	94.17	**95.63**	90.70
Aston- Revolution und Contrerevolution	91.19	92.58	**92.71**	92.06
Auerbach- Der Lehnhold	87.11	**92.34**	90.70	89.25
Balzac- Vater Goriot	94.71	**95.91**	94.88	94.37
Bierbaum- Ein Roman aus der Froschperspektive	93.43	93.11	**94.89**	91.91
Bruckbräu- Mittheilungen aus den geheimen Memoiren einer deutschen Sängerin	89.38	**93.36**	92.74	90.03
Duncker- Großstadt	89.25	92.97	90.86	**93.28**
Fischer- Gustavs Verirrungen	93.59	93.41	**93.66**	92.70
Fontane- Quitt	89.15	**91.87**	89.60	83.72
Fontane- Stine	93.35	**95.0**	93.16	94.16
Franzos- Der Pojaz	88.69	**93.82**	91.54	93.43
Klabund- Bracke	83.30	**89.94**	89.82	88.99
May- Auf fremden Pfaden	**91.00**	89.84	88.59	87.51
Pichler- Agathocles	85.51	**87.39**	86.92	85.39
Sack- Paralyse	87.16	**89.20**	88.20	88.01
Verne- Zwanzigtausend Meilen unter'm Meer	92.02	**93.17**	93.08	91.16

the segment contains some dialectial paragraphs. However there are some cases where the algorithm fails to detect a character reference on seemingly easy tokens such as *"Vater" (father), "Mutter" (mother) or "Herrn" (Mister).* In *Quitt*, Fontane used terminology related to the english military which is not detected by the algorithm. The Precision, even though it is quite high already suffers from false positives originating from the detection of *"Fort Holmes"* which happens several times. In *Bracke* the algorithm shows a very high Precision, showing only a weakness in a section which is centered around a *mace* named *"Ananke"*. The Recall however is much lower than in other documents. The missing terms are mostly appellatives, ranging from simple salutations to noble titles, many of which have been detected by the rule-based approach. In *Auf fremden Pfaden*, the algorithm shows a Precision of almost 97% and beats both the rule-based and the MaxEnt classifier in the F1-score. Mistakes are largely caused by missing Arabic names entirely or not recognizing the full composition (e.g. *Ben Sakir* instead of *Abram Ben Sakir*). *Agathocles* shows similar characteristics but instead of Arabic names the document deals with Roman ones, such as *Valerian* or *Agathokles*, which were not captured. Aside of a single false positive, all false positive are due to pronouns, which refer to concepts such as *"Tod" (death)*, which can be referred to by male pronouns in German.

Rule System The rule-based system shows severe weaknesses in two passages in *Auerbach*, that are not centered around human characters but animals instead. The first of those causes about 15% of all false positives (131 in total). The false negatives are made of pronouns as well, one passage is centered around a child, which can be referred to by *"es" (it)*, which is generally excluded in the detection. The appellatives which were not detected are seldom used in German texts such as *"Niederen", "Niedersten" or "Preiswürdigen"*. In *Quitt*, the rule-based system has severe problems with locations, especially everything which follows *"Fort"*, such as *"Fort MacCulloch" or "Fort Holmes"*. There are four locations which are responsible for close to 30% of all false positives, which is the main reason for a low Precision of just 77.7%. The text *Bracke* shows a rather philosophical nature and uses terms which usually refer to character references such as *"Mutter" (mother) or "Schwester" (sister)* as abstract concepts. On top he creates many rather unusual compound words which are hard to detect. There are two main problems in *Auf fremden Pfaden*. The first is that the rule-based system does not detect Arabic names very well (e.g. *"Sihdi" or "Tuareg"* and the templates which build entities from the individual tokens are not adapted to constructions such as *"Hadschi Kara Ben Nemsi"*. The novel *Agathocles* features mainly roman characters in the style of a romantic letter. The main figures appear mainly at the beginning and the end of those letters, but the current rule-based system does not include dedicated rules for these instances in its current state (e.g. one could treat each letter similar to a direct speech).

MaxEnt + Rules In *Auerbach*, the system has the same weakness as the rule-based system, it cannot detect passages that focus around an animal and classifies pronouns as character references. The false negatives are mainly due to unusual compound words such as *"Domänenraths" or "Oberamtmännin"* which is rather specific vocabulary. For

Table 6.11.: Correlation coefficients between the different approaches. All approaches show a positive correlation and all machine learning approaches are strongly correlated.

Approaches	**MaxEnt**	**Rules**	**BiLSTM-CRF**	**E2E**
MaxEnt	-	0.79	0.86	0.76
Rules	-	-	0.71	0.60
BiLSTM-CRF	-	-	-	0.80

Quitt the algorithm handles locations much better than the rule-based system but has some issues with the term *"Francisco"*. The remaining errors in this document are very specific. The issues in *Bracke* with abstract reference remain, however the other specific compounds are handled slightly better. The results on *Auf fremden Pfaden*, written by Karl May, shows exactly the same characteristics as the other systems, namely a lot of Arabic names that can either not be found or not arranged in the correct phrase. Same issues with *Agathocles*, the Roman names are not detected which explains a rather low Recall.

BiLSTM-CRF In *Auerbach*, the system fails to detect the passages centered around non character references, but especially fails to detect many specific appellatives and even names such as *"Ameile"* or *"Vreni"*. The system shows the same problems in *Quitt*, *Agathocles* and *Bracke* as the other systems with many mistakes due to some rather specific appellatives that were not found. There is one characteristic worth noting in *Auf fremden Pfaden*. Similar to the other systems, the CRF has huge issues finding the correct span of the Arabic names, which is surprising since the CRF is specifically designed to handle label sequences.

Summing up, the issues that appear throughout the different systems are rather similar. Romantic letters need specific features, sections that deal with entities that do not refer to a literary character need to be identified, as well as both dialects and name schemata prevalent in different regions on earth have to be considered. We are giving the Pearson correlation coefficient between the different systems in table 6.11. The systems all show a positive correlation between each other, with the highest correlation occurring between the MaxEnt and the BiLSTM and the lowest between the rule-based system and the neural ranking system.

6.10. Generalization Capabilities on the Kindler data set

This section examines the generalization capabilities of different approaches towards the Kindler data set, introduced in section 6.3. The training data remains the same as in the previous section, that is 72 documents of the DROC data set. The results are shown in table 6.12.

Adding rules as features to the classical approaches is still beneficial, however the quality drops from almost 93% down to just about 86% for the unlabeled setting and

Table 6.12.: Results on the Kindler data set, all approaches perform much worse compared to in domain data, but the drop is rather constant across classifiers.

Experiment	$F1_{ul}$	$F1_{lab}$	$F1_{core}$	$F1_{app}$	$F1_{pron}$
Rules	84.89	71.38	62.33	62.05	83.76
MaxEnt +Rules	85.30	**74.10**	68.32	66.55	**83.84**
Perceptron$_{Mira}$ +Rules	**85.63**	74.03	**68.90**	**66.74**	83.29
MaxEnt	83.26	72.70	66.85	65.56	82.46
Perceptron$_{Mira}$	83.39	72.56	67.38	65.55	82.06
BiLSTM-CRF	79.81	69.81	60.41	61.38	81.73

from about 90% to just about 74% in the labeled setting. The quality of the rule-based approach beats the Perceptron and the MaxEnt on the pure detection of the spans (unlabeled) but since one of its weaknesses is the determination whether a span is an appellative or a proper noun, it is outperformed on the labeled setting. The difference is most significant for the BiLSTM-CRF, with a labeled drop of about 12% and an unlabeled drop of almost 20% F1-score.

6.11. Discussion

In this work, we examined four different approaches to obtain a character reference detection system for German literary texts. Two of those approaches do not strictly rely on labeled data in the target domain (the rule-based system and the transfer learning approaches). We started with the baseline CRF of Stanford which yields an F1-score of only about 8% when evaluated on the test section of the DROC corpus. By extracting gazetteers from a large source of unlabeled documents, and using them to annotate common nouns and pronouns, about 72% F1 of all spans could be detected. We then proceeded to use these gazetteers to label the 90 documents of DROC with weak labels and retrained the CRF of Stanford and obtained a score of almost 78% F1 using this scenario, usually referred to as *distant supervision.* Integrating Expectations into previously released corpora based on newspaper texts and Wikipedia articles improved the classification score to more than 82% F1, outperforming the baseline in all three categories.

The use of Active Learning showed that by labeling about 10.000 tokens an unlabeled score of about 92% can be obtained, increasing very slowly from this point onwards. The main cause for the quick rise is that the classifier learned to correctly classify pronouns which dominate the references with a proportion of about 66%. I propose to make use of Active Learning for CRD by training a classifier using Active Learning and to use this classifier to preprocess entire documents and revise this labeling manually afterwards.

Our rule-based approach reached about 91% F1-score in the unlabeled setting but still has severe weaknesses when differentiating between proper nouns and appellatives. A second issue with the rule base system is that - at the current point in time - it has no dedicated mechanism to differentiate between personal pronouns referring to characters and those that do not.

If labeled data is assumed, we experimented with unstructured and structured classification as well as ranking systems. The results suggest that feature based systems are still on par with BiLSTMs that only receive embeddings as their input.

The integration of the rule-based systems into the feature based classifiers improved the unlabeled score by almost 1%, reaching a classification score of close to 93%. The correct spans of proper nouns could be detected with more than 85% success rate and appellatives seem to be the hardest category with a score of about 82%. The highest score for the detection of pronouns that refer to character references is given by the BiLSTM-CRF with a score of more than 93%.

The error analysis showed that the systems have similar issues but some systems can deal with these issues more efficiently than others. In total, the results of the systems on 18 test documents correlated in the Pearson correlation coefficient. When applying these new systems onto a different domain, the classification score drops significantly by more than 15% labeled F1, showing that the problem of domain adaptation still remains.

This chapter showed, that by the usage of generalized expectations, a shift in the task and the domain can be tackled efficiently. But even this approach falls short when compared to pure supervised algorithms, which show a further increase in the F1 score of 10%. The creation of a rule-based system still offers a possibility to get access to an algorithm of good quality (close to the results of purely supervised systems) but requires a domain expert. If high quality is the main goal, there is no way to avoid the annotation process, but we propose to make use of a semi-automatic process, as depicted in section 6.3.

7. Attribution of Speaker and Addressee to Quotations and Dialogs

[1]In the literary domain, the resolution of speaker and addressees is a vital part for the Coreference Resolution. The DROC corpus contains about 29 % of sentences that are located inside direct speech and about 20.000 of our annotated 50.000 references are located inside direct speech (DS). The direct benefits of speaker and addressee detection for CR are as follows:

1. All pronouns located inside DS in first person singular can be resolved directly to the speaker.
2. All pronouns located inside DS in second person singular can be resolved directly to the addressee.
3. Pronouns in third person are highly unlikely to corefer to either speaker or addressee.

Furthermore, their detection opens up the possibility to create character networks of literary novels [Elson et al., 2010] which model interaction based on communication between entities and nodes in the form of speaker/addressee entities. The next section provides an overview over the task, and its main challenges, followed by the description of our current rule-based system for the detection. After the description of our rule-based algorithm, we provide results with machine learning in section 7.4, both for speakers and addressees.

7.1. The task and challenges of speaker detection

Beginning with the guidelines for the annotation of speaker and addressees in DROC, this section tries to conceptualise the task of speaker detection so that subsequent algorithms can be designed to solve this task.

Guidelines for the manual annotation of speaker and addressee

For DROC, for every direct speech utterance, one or more references out of all character references were selected to act as the speaker/addressee of this DS (This means that the portion of speaker is a subset of all annotated character references). A direct speech in DROC is just a continuous segment of text, usually but not necessarily enclosed in

[1]The major part of this chapter is submitted for publication [Krug et al., 2019b]

quotation marks.

The choice, which references can act as speaker or addressee, is restricted, so that:

- No reference inside a DS can act as a speaker.
- No pronoun inside a DS can act as addressee.
- when no *frame* of a DS is available, then the last reference of the according entity is selected as speaker/addressee.

A frame in this manner is an optional phrase of text (not annotated) introducing a direct speech utterance which most of the times contains the according speaker (sometimes the addressee is mentioned as well). The frame can either be located in front of an utterance, in between two utterances, or can follow directly after an utterance, an example for these categories is given in figure 7.1.

... **als sie plötzlich, nach dem Fenster zeigend, ausrief:** »Mein Gott, was ist das? ...[a]

»Richard« – **sagte sie leise.** – »Geliebter! warum fliehst Du mich? [b]

»Sie liebt die Einsamkeit, gnädige Frau« – **erwiederte die Gesellschafterin** ...[c]

»Unbegrenztes, mein Richard.« »Nun, also –« »Heute zum Beispiel ...«

[a]**when suddenly, pointing to the window, she cried out:** ”My God, what is that? ...
[b]”Richard” **- she said quietly. -** Beloved! Why do you flee me?
[c]”She loves loneliness, madam” **- the partner replied ...**

Figure 7.1.: Examples for Direct Speech utterances (all taken from Aston Louise - *Lydia*) with a frame prior to the utterance, in between two utterances, after the utterances and one utterance without any frames. The frames are marked in bold

These guidelines were selected so that no trivial resolutions of pronouns (such as using ’ich’ (I) or ’du’ (you) as speaker or addressee) inside the DS can solve the task. This annotation scheme supports our primary goal to be as beneficial for coreference resolutions as possible.

Conceptualisation of the task of speaker detection

If a frame is present (66% of all cases), the speaker is usually mentioned explicitly and therefore the task becomes to choose the best reference inside the frame for the speaker. This is an instance of a ranking problem. Whenever no such frame is available, the

»Ist der Baron von Landsfeld hier?« war ihre erste Frage an **Herrn von Maienberg** [a]

»Was schreibt er Dir denn, **Lydchen**?« fragte die neugierige Alte[b]

[a]"Is Baron von Landsfeld here?" was her first question to **Mr. von Maienberg.**
[b]"What does he write you, **Lydchen**?" asked the curious old woman

Figure 7.2.: Examples for utterances with explicit addressees. There are two different possibilities for an explicit mention. The first one is when the addressee is located in an adjacent frame and the second one is when the addressee is located inside the utterance. The addressees are marked in bold.

DS is usually embedded into a dialogue of one or more entities and the task is now to propagate the most likely candidate from adjacent DS utterances.

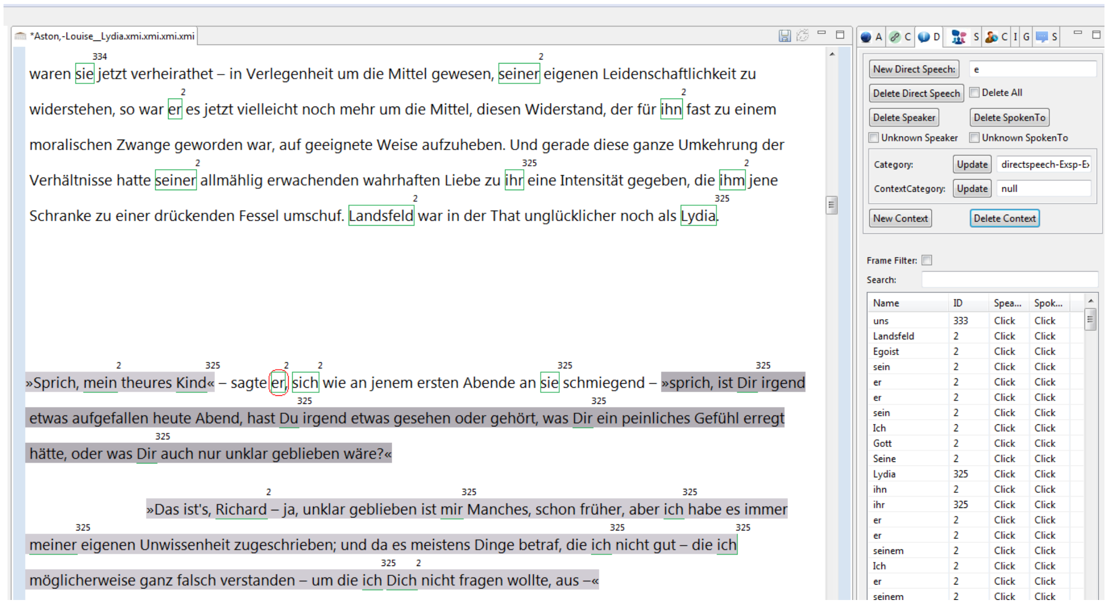
This represents the core idea for our rule-based approach, to detect the references inside frames that act as speaker and then propagate them to adjacent ones. For the experiments involving machine learning we tried both, to have the algorithm deal with both cases in the same pass, as well as to predict just the ones explicitly mentioned in the frame.

The detection of the addressee turns out to be different. Only in very few instances (about 25 % of all utterances), the addressee can be found in a frame of the DS (see figure 7.2).

If there are hints for the addressee, then they are usually located inside a DS, but for the majority of DS, no hints are given and the addressee has to be inferred from nearby utterances. This lack of explicit information is the reason why our rule-based approach only searches an addressee in the utterance and if no such reference can be detected then it has to be inferred from nearby utterances. The lower percentage of explicit mentions of addressee (about 33 %) compared to speaker (with about 66 % of all utterances having the speaker located in a nearby frame) also shows that the detection of the addressee is more challenging. The next section shows the annotation of the speaker and addressees using ATHEN.

Annotation and Text Highlighting ENvironment (ATHEN)

This section introduces the features of ATHEN [Krug et al., 2018b] (see chapter 5) that were used for the annotation of the data set used in this work. ATHEN offers a feature to preprocess incoming documents with a preconfigured pipeline based on Apache UIMA that results in an automatic annotation of character references. If the goal is to annotate both - the mentions as well as their coreferential information, the direct speech utterances and their speaker information - then we opted for a two pass approach. In the first pass through the text, the user can correct all suggested mention spans and assign the according coreferential ID (we restrained from annotating coreferential edges and



waren sie jetzt verheirathet – in Verlegenheit um die Mittel gewesen, seiner eigenen Leidenschaftlichkeit zu widerstehen, so war er es jetzt vielleicht noch mehr um die Mittel, diesen Widerstand, der für ihn fast zu einem moralischen Zwange geworden war, auf geeignete Weise aufzuheben. Und gerade diese ganze Umkehrung der Verhältnisse hatte seiner allmählig erwachenden wahrhaften Liebe zu ihr eine Intensität gegeben, die ihm jene Schranke zu einer drückenden Fessel umschuf. Landsfeld war in der That unglücklicher noch als Lydia.

»Sprich, mein theures Kind« – sagte er, sich wie an jenem ersten Abende an sie schmiegend – »sprich, ist Dir irgend etwas aufgefallen heute Abend, hast Du irgend etwas gesehen oder gehört, was Dir ein peinliches Gefühl erregt hätte, oder was Dir auch nur unklar geblieben wäre?«

»Das ist's, Richard – ja, unklar geblieben ist mir Manches, schon früher, aber ich habe es immer meiner eigenen Unwissenheit zugeschrieben; und da es meistens Dinge betraf, die ich nicht gut – die ich möglicherweise ganz falsch verstanden – um die ich Dich nicht fragen wollte, aus –«

New Direct Speech: e
Delete All [ ]
Unknown Speaker [ ]
Unknown SpokenTo [ ]
Category: directspeech-Exsp-E
ContextCategory: null
Frame Filter: [ ]
Search:

Name	ID	Spea...	Spok...
uns	333	Click	Click
Landsfeld	2	Click	Click
Egoist	2	Click	Click
sein	2	Click	Click
er	2	Click	Click
er	2	Click	Click
sein	2	Click	Click
Ich	2	Click	Click
Gott	2	Click	Click
Seine	2	Click	Click
Lydia	325	Click	Click
ihn	2	Click	Click
ihr	325	Click	Click
er	2	Click	Click
er	2	Click	Click
seinem	2	Click	Click
Ich	2	Click	Click
er	2	Click	Click
seinem	2	Click	Click

Figure 7.3.: The DirectSpeech-View of ATHEN: The left side shows the editor with the text that is currently worked on. Each text segment highlighted by a box shows a character reference, with the according entity ID written at the top right of the snippet. All references that have the same ID refer to same entity. The text snippets with a solid background show direct speech utterances, and the reference that is circled marks the speaker for the currently selected direct speech instance. The right side shows the view itself, which offers functionality to select a speaker/addressee from a table containing all visible mentions as well as functionality to add, delete or edit direct speech utterances.

modelled an entity simply by assigning a cluster ID to every mention), and during the second pass over the text they can correct all proposed direct speech utterances and assign the according speaker and addressee from the previously annotated mentions. Figure 7.3 shows the so called *DirectSpeech-View* of ATHEN. The centre shows the editor with the document that is currently worked on. The document shows all mentions, depicted with boxes, as well as all utterances, depicted with a grey background, and the currently assigned speaker for the direct speech that is selected by the caret in the editor (depicted as a circle around a mention).

7.2. Related Work

One of the first publications in this field is the ESPER system [Zhang et al., 2003], which is designed to recognise direct speech utterances within children's stories. The system first extracts the direct speeches in the text and classifies them with a decision tree into two categories, speaker change or no speaker change. The results are evaluated with two very different, manually annotated stories. They report the quality in accuracy, which is the ratio of correctly determined speakers for all direct speeches: 47.6 % and 86.7 % for the two stories. Glass and Bangay [Glass and Bangay, 2006], who developed another rule-based system, first determine the communication verb for a direct speech and then a set of candidate actors from which the speaker is finally determined. They evaluate their techniques on 13 English fictional works and report an accuracy of 79.4 % [Glass and Bangay, 2007]. Iosif and Mishra [Iosif and Mishra, 2014] follow in principle the scheme of Glass and Bangay, but supplement it with a more elaborate pre-processing, including a reference resolution. They achieve an accuracy of approx. 84.5 % and are therefore among the best published results to date. Ruppenhofer et al. [Ruppenhofer et al., 2010] report an F1-score of 79 % in the attribution of politicians to their statements in German cabinet protocols from the years 1949-1960.

In addition to these rule-based approaches, machine learning methods are also used. One of the first successful systems using machine learning is that of Elson and McKeown [Elson and McKeown, 2010]. They labelled their speaker assignment data via Amazon's Mechanical Turk System. Their system first classifies a direct speech into one of five syntactic categories based on rules. Independent machine learning methods were then trained for each of these categories. In total, they achieved an accuracy of about 83 %, evaluated on the basis of English novels from the 19th century. O'Keefe et al. [O'Keefe et al., 2012], who criticise Elson and McKeown's approach because of the way they created the gold standard and also because of the use of information from the gold standard in application[2], regard the assignment as a sequence problem. They use the classification information from previous direct speeches as features for the entire sequence. In their evaluation, they compare three methods with a very simple rule-based baseline. Their results in applying the system to two newspaper corpora - the Wall Street Journal and the Sydney Morning Herald - as well as the collection of literary texts from the work of Elson and McKeown show a big difference between the domains. They achieve 84.1 % (WSJ) and 91.7 % (SMH) accuracy on the two newspaper corpora. On the literary corpus, however, they achieve a maximum accuracy of only 49 %. He et al. [He et al., 2013] achieve an accuracy between 74.8 % and 80.3 % on the Elson and McKeown corpus with a ranking based machine learning method using features of the Actor-Topic Model [Celikyilmaz et al., 2010]. Almeida et al. assume a close interweaving of coreference resolution and speaker attribution and thereby integrate both procedures in their approach; the results of the two individual learning procedures are combined in a third step using Integer Linear Programming. They thus achieve 88.1 % accuracy [Almeida et al., 2014]. The only reported experiments using deep learning methods, based on the language of the figures, have achieved accuracies of less than 50 % [Chaganty and Muzny, 2014].

The identification of addressee is also referred to as the detection of listeners of utterances [Yeung and Lee, 2017]. They detect both the speaker and the addressee using a single CRF classifier. For each direct speech utterance, they used two sentences be-

[2]During prediction of speakers in a dialogue, the gold speaker of previous direct speech utterances were used as features

... und seine Freude, die er über dies »Geheimniß« zu erkennen gab, war diesmal aufrichtig. [a]

»Es ist die höchste Zeit« – dachte er. [b]

[a]... and his joy, which he revealed about this »secret«, was sincere this time.
[b]»It is about time« - he thought

Figure 7.4.: Examples for utterances that do not depict direct speeches. The first one is labelled as *citation* and the second one is a *thought*

fore and afterwards as context and labelled each token as either 'speaker', 'listener' or 'neither' They tested their algorithm on two data sets, the novel *Emma* and the *New Testament* and achieved an accuracy of 52.5 % and 66.1 % for the speaker detection, as well as 28.5 % and 57 % for the detection of the addressee.

7.3. Rule-based speaker detection

As already introduced in the previous section, the automatic detection of speaker and addressee for the literary domain can be separated into a two-step process:

1. Resolution inside frames (*Explicit Resolution*)
2. Propagation from adjacent DS utterances (*Dialogue Propagation*)

This section describes the current state of our rule-based system for the resolution of speaker and addressee. Both steps require their own data structures to operate on. The Explicit Resolution process relies on the DS utterances and their immediate proximity. In this manner, each DS is modelled as an object with a list of frames (which could be empty, the frames have to be detected in the first step) as well as two separate lists of candidates for the speaker position and the addressee position respectively. As candidates for speaker we select all references that are located in the according frames of a DS. The candidates for the addressee position have to be located inside the DS utterance and may not be pronouns.

The current implementation of this algorithm expects character references as well as a list of direct speech utterances as its input.

Even with just the DS that are surrounded by quotation marks there is another challenge for our rule-based system to overcome before the actual speaker detection, namely to detect whether the enclosed segment represents a direct speech or a different category. Figure 7.4 shows different segments that are marked the same way as a DS but represent a different category, such as a citation, thoughts or passages of songs or writing.

Since this could confuse the algorithm in a later step, we developed a scheme to filter those utterances before they were used for speaker detection. This process is referred to as *Utterance Category Determination* and explained in the next section alongside the other steps of the algorithm.

Utterance category determination Given a list of utterances, the task is to assign a label for each of them. While the current implementation would allow a more fine-grained analysis, for the process of Speaker Detection it is sufficient to just assign two labels, namely *directspeech* and *other*. For the determination of non-directspecch utterances, we use six heuristics:

- A *Non-speaking relation* has been detected in the frame of the utterance. (This is determined by examining the dependency paths of the candidates and the verbs inside the frames of the utterance)
- There is a noun phrase reaching from outside the utterance right into the utterance.
- The utterance is embedded in another one (they have to be treated different than the rest of the DS utterances, more precisely they will be resolved in a separate pass of the algorithm.)
- The first token inside the utterance starts lowercase.
- Sentence heuristic, which is evaluated to true, if there is a segment preceding and following the utterance.
- The utterance has a preceding frame with a ':' that is introduced with a list of special indicator phrases.

Whenever one of those heuristics is evaluated to true, then the category of the utterance is changed to *other*, all remaining utterances are labelled with directspeech and therefore serve as the input to the speaker detection algorithm.

Detection of frames The first step of the speaker resolution procedure is to detect the frames for a DS. Since a frame is an adjacent snippet of text, our algorithm detects a frame as follows: At the beginning, the first token prior/after the utterance is selected, and then extended until either a sentence boundary or a token which is located in another DS utterance is found. This means, we extended the DS by its directly surrounding sentences, which are only present if no DS is adjacent. Then, for those sentences an *affinity* to the DS is calculated. If the sentence following the DS starts with a lowercase token or one of the characters *,-–;*, then the frame gets an affinity of *high*, that is we can expect to find a speaker in this frame. For the sentence prior to the DS, the last token needs to be one of the characters *:,;–* to be determined as a frame with high affinity. All other frames get a *low* affinity to the DS assigned.

Explicit speaker resolution For both, the explicit resolution of speaker and the explicit resolution of addressee, we implemented a rule-based system based on a simple scoring mechanism. Each candidate gets examined to be in different positions and whenever one of these is met, a score is assigned to the candidate. In the end, the candidate with the highest score gets assigned to be speaker of a DS. The scores were chosen by intuition and were not determined on the training data and have to be interpreted in a way, that a positive score awards a candidate to be speaker while a negative score shows that it is possibly not a good choice to select this candidate as speaker. The features for the explicit speaker resolution are as follows (all rules work only for candidates that are located in a frame with high affinity towards the DS):

- If the relation between the candidate and the verb depicts a *speaking relation* (e.g. ', sagte er' (', he said')) (+0.3 score)
- If there is only a single candidate in the frame (+0.1 score)
- If the candidate is in a nominal noun phrase (+0.1 score)
- If the candidate is detected to be a subject) (+0.05 score)
- There is no comma in between the candidate and a potential colon, preceeding a DS (+0.4 score))

Alongside these rules, there is a fallback rule (which is applicable to candidates in frames with low affinity as well) that adds a small score of 0.02 if there is only a single candidate for a DS. Reciting the features one may notice that there are no negative scores. While formulating rules with a negative impact is relatively easy per se (such as: Possessive Pronouns are unlikely to speak; or that both genitive and dative positions are unlikely to speak) we found that there were either no conflicts with positive scoring rules, yielding no improvement or that the information was only present with low reliability (especially dative constructions are detected poorly by the parser). The results of the explicit detection are given in section 7.4.

Explicit addressee resolution Similarly to the explicit speaker resolution, for each candidate which is located inside a DS, scores were given for:

- The candidate is surrounded by *non words* (e.g. sentence delimiters, quotation, commas,...) (score +0.4)
- The chunk (as determined by the TreeTagger [Schmid, 1995]) is surrounded by non words (score +0.4)
- The chunk contains any articles, conjunctions or indefinite pronouns (score -0.45)
- The chunk contains possessive pronouns in third person (score -0.45)
- The candidate is to be found in plural position (score -0.45)

These rules form the backbone of the algorithm. There are just two rules being able to yield a positive score, however they generate too many false positives and therefore the three negative conditions prevent the over generation by setting the score to a negative value. At the current state, there are still situations, where the rules predict false positives, however the resolution would require further information by the parser which is very unreliable inside the vocabulary of a direct speech (e.g. appositive constructions can not be filtered, also there should be no relative pronouns referring to the candidate)

Creation of non-continuous direct speech utterances After all DS have been analysed towards their potential to contain a speaker/addressee which is mentioned explicitly, the next step consists of the creation of non continuous utterances (which we call *splitDS*. The idea behind this step is that splitDS usually share the same speaker and addressee. In order to create them, multiple DS get merged to form a single splitDS. This step is done by exploiting syntactic properties as well as context specific consistencies found in the layout of the document. The first rather easy property used here is to combine utterances which are embedded in a single sentence. Depending on source and formatting, some documents share the property that all utterances that are mentioned inside the same paragraph (a paragraph is a continuous text snippet visually separated by line breaks) are spoken by the same mention and directed towards the same set of references. We found that if paragraphs are available, then a single rule can be applied to combine adjacent DS, which combines all utterances of a single paragraph into a splitDS. This usually combines just two DS with each other.
Detecting and making use of this consistency is rather important. The entire algorithm is based around the concept to start from the easier and smaller utterances, attribute them with their according speaker and addressee and afterwards it tries to resolve the speakers on the dialogue level. Since a dialogue is usually based around multiple entities talking in turn it is important to detect and rule out any utterances that would break this property.

Dialogue level resolution This step can be considered the second pass of the algorithm in the sense that it makes use of all previously detected speaker and addressees and tries to propagate this information to nearby DS. The first task for this stage is to determine dependencies between DS. More precisely the algorithm finds the DS that are very likely to be part of the same dialogue of the story. All utterances are partitioned into *dialogues* by examining, whether their assigned frames have an overlap. Since there are small intermediate passages in between utterances that do not indicate the end of a dialogue, those smaller dialogues are merged into one, when there is no more than one sentence in between them. The structure of the resulting dialogues is then examined further. In order to do this, among all references inside this dialogue (excluding the references inside the utterances) we applied our adaptation of the Stanford sieve coreference algorithm (see chapter 8). This provides us with the information, how many entities are involved in this dialogue (however this information is not entirely reliable). The explicit speaker or addressees that were detected in previous stages are now propagated to adjacent utterances. This is done by a set of local rules (an example rule is that whenever the speaker of one utterance was determined to be coreferent to the addressee of the following utterance, and only the addressee of the first utterance is missing, then we propagate the speaker of the second utterance). In total there are currently 16 rules of this kind that are applied one after another. If after this procedure, there are still utterances without speaker/addressee, then we assume that we can simply propagate them in an alternating fashion, as long as no two adjacent utterances

Table 7.1.: Results of the rule-based system on the detection of speaker and addressee on train and test data respectively.

Approach	Accuracy	Precision	Recall	F1
Speaker (train data)	84.6	88.7	84.9	86.8
Addressee (train data)	59.5	68.7	61.5	64.9
Speaker (test data)	84.8	88.0	85.0	86.4
Addressee (test data)	65.3	73.3	67.3	70.1

share the same speaker entity (as determined by the coreference resolution). This mechanism can be considered a default mechanism.

The results of the rule-based system for the detection of the speaker and the addressee are depicted in table 7.1.

The rule-based system yields an accuracy of about 85 % for the speaker and about 65 % for the addressee. These results are rather similar to [Krug et al., 2016] when the available data set was still in creation. The detection of the speaker appears to be easier than the prediction of the addressee. This is probably due to the speaker being mentioned explicitly more often (about 66 % of the time) compared to the addressee (about 25 % of the time). A separate evaluation of the explicit detection of the algorithm (which means the evaluation only applies the explicit stage of the algorithm and only evaluates on data relevant for this stage, see section 7.4) shows that the explicit speaker can be detected with a success ratio of about 93 % while the accuracy for the explicit detection of the addressee only yields an accuracy of about 65 %. Considering this, the propagation step conserves the quality for the addressee while reducing the quality of the explicit resolution for the speaker by almost 8 %. To see, how this approach compares with state of the art machine learning approaches, we performed according experiments and reported the results in section 7.4.

7.4. Speaker detection based on machine learning

We extended previous experiments using machine learning [Krug et al., 2016] on DROC. At the time of publication, the corpus was still in creation and therefore featured less annotations for training and evaluation. On top, different settings and features have been integrated into the classification setting. We reexamined the two best performing setups on the completed corpus and also compared it to modern deep learning approaches. We start with the description of the detection of the speaker.

The first setting that is examined (called '2-Way') is built as follows: For each direct speech utterance, k character references that appear closest to the utterance are selected as candidates. An instance for classification is created for each tupel consisting of the utterance and one of the candidates. The job of the classifier is then to predict whether the candidate is the speaker or not. The amount k as well as how the *closest* candidates are defined has to be determined on the validation set. During classification, the candi-

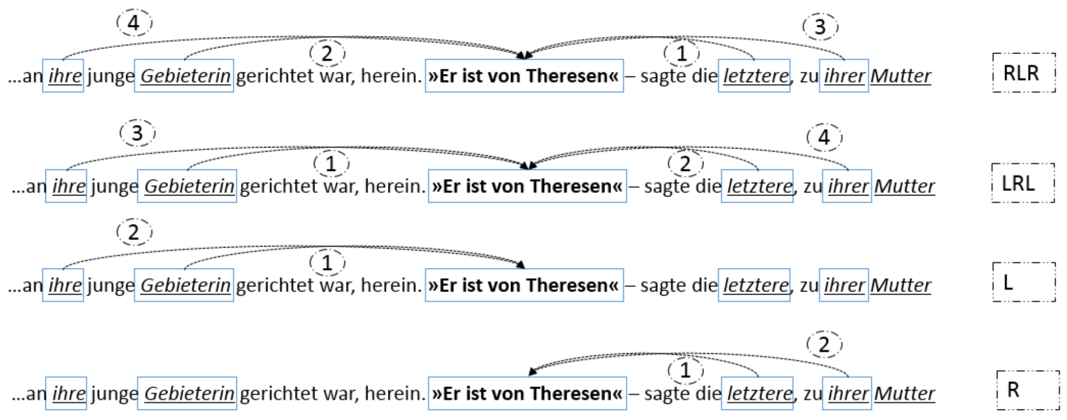

...an ihre junge Gebieterin gerichtet war, herein. »Er ist von Theresen« – sagte die letztere, zu ihrer Mutter
RLR
LRL
L
R
1
2
3
4

Figure 7.5.: The four different strategies for the pairing (and the order) of candidates with a direct speech utterance. R looks for candidates on the right and L for candidates to the left of the utterance. The numbers close to the candidates depict the order in which the are presented to the classifier.

date with the highest (positive) scoring gets selected to be the speaker. If no positive label is found, then no speaker could be determined.

One drawback of such an approach is that this introduces a hyper parameter with a large range of possible values. One possible solution is to apply a trick similar to the sampling method for coreference resolution introduced by Soon et al. [Soon et al., 2001]. Instead of setting k to a fix value during an experiment, instances are created until a positive candidate is found. This way the hyper parameter can be reasonably reduced from a positive integer to just four different values (We refer to this strategy by the term '...tillMatch'). The four different values are named *Right*, *Left*, *Right-Left-Right* and *Left-Right-Left* and address the order in which candidates are paired with the utterance and presented to the classifier. An example of these strategies is shown in figure 7.5.

Both approaches combine all different steps of the speaker recognition into a single model. That is, the model has to detect speakers that are propagated from nearby ones, has to detect if the utterance is a speech or a citation and has to detect explicit speaker in nearby frames of the utterance. For this stage, we experimented with a Maximum Entropy classifier and a Perceptron, both using the feature set depicted in table 7.2.

The results of these experiments are depicted in table 7.4, with the explanation of the performance metrics given in section 7.4. For the entries of this table, we tuned the hyper parameters (windows size for the 2-Way strategy and pairing order for the tillMatch strategy) on the validation set and used the best constellation for evaluation on the test set. We provide details about the influence of the individual parameters in table 7.3.

In both cases, the *RLR* strategy turned out to be the best performing one, however the different classifier had a major difference when it comes to the best window size. The

Table 7.2.: Features used for the classical supervised machine learning approaches for the detection of the speaker.

Feature name	Description
Subject, dative, accusative	Whether the candidate is in a subject, dative or accusative object position
Distances	The distance in tokens to the utterance as well as the distance in character references till the utterance
Tokens Between	The token, POS Tags and verbs between the candidate and the utterance
GermaNet between	The semantic field category in GermaNet of all tokens in between the candidate and the utterance
Dependency	All tokens, POS Tags and lemmas of the dependency path (till the root node)
Context type	All features above paired with the information what kind of frame the candidate is located in (no frame, previous frame, or following frame)

Table 7.3.: Results of the classifier for the detection of speakers on the validation set, listed with different strategies for the creation of the candidates. While the *RLR* strategy performs the best, just searching after the utterances yields very high Precision values.

Classifier + Strategy	**Accuracy**	**Precision**	**Recall**	**F1**
MaxEnt + LRL	63.3	63.3	66.2	72.3
MaxEnt + RLR	64.5	80.2	67.3	73.2
MaxEnt + L	37.5	74.0	39.0	51.1
MaxEnt + R	43.7	88.4	44.1	58.8
Perceptron + LRL	65.6	80.0	68.7	73.9
Perceptron + RLR	65.6	79.8	68.6	73.8
Perceptron + L	37.7	73.9	39.1	51.1
Perceptron + R	44.2	89.9	44.5	59.6

Table 7.4.: Results of machine learning approaches for the detection of speakers on the test set, with their best hyper parameters tuned on the validation set. The reported accuracy and other metrics are reported in %.

Approach	**Accuracy**	**Precision**	**Recall**	**F1**
MaxEnt2Way, window=13	66.3	70.8	70.1	70.4
Perceptron2Way, window=5	69.1	72.7	73.0	72.9
MaxEntTillMatch, RLR	66.0	76.3	67.7	71.8
PerceptronTillMatch, RLR	66.6	75.7	68.3	71.8

MaxEnt classifier had the most benefit from a bigger window size while the Perceptron performed best with a window size of 5 candidates to each side.

The results for this setup of machine learning yield a maximum accuracy of close to 70 % on the test set (using the MIRA Perceptron), which is about 15 % lower than the score of the the rule-based algorithm. The 2-Way strategy slightly outperformed the tillMatch strategy and the Perceptron proved to be a bit more robust compared to the Maximum Entropy classifier.

Since we did not expect to be able to improve the score by more than 15 % to be at least on par with the rule-based system just by changing the features or switching to deep learning methods, we restricted the task, so that the machine learning algorithms were not forced to predict every speaker during a single prediction step. This restriction can be motivated by observing the trend of the Precision in figure 7.6, when increasing the window size for candidate generation. Especially when small window sizes are applied, the algorithm can correctly predict the speaker for the utterance. This might be caused by the detection of speakers in the according frames (the algorithms have to detect the frame information implicitly). In order to examine this effect, we proceeded with the restriction that the algorithm should now only predict a speaker for direct speeches in which frames could be detected (using our rule-based approach of detecting frames with high affinity). This setup is referred to as *frame*-speaker detection. Not only does this setup help to get rid of the aforementioned hyper parameters entirely, the task is also handled more intuitively and similarly to the rule-based approach. The downside of this approach is that the algorithm cannot predict a speaker for every utterance and there has to be a second propagation step applied afterwards. Table 7.5 shows the fraction of direct speech utterances where a frame could be detected by the rule-based algorithm. In the test set, these 484 utterances are paired with about 1000 candidates yielding a baseline of about 62 % for the correct assignment of speaker to direct speech utterances.[3]

We experimented with traditional feature based classifiers (Maximum Entropy and Perceptron), one ranking approach and two deep learning architectures. The first architecture encodes the internal state using LSTM-cells, and the second architecture uses Transformer layers, see section 7.5 for an in detail description of the individual model and their parameters.

[3]The baseline is not 48% as one would expect but higher, because any selected candidate, which does corefer to the correct entity is considered a successful resolution.

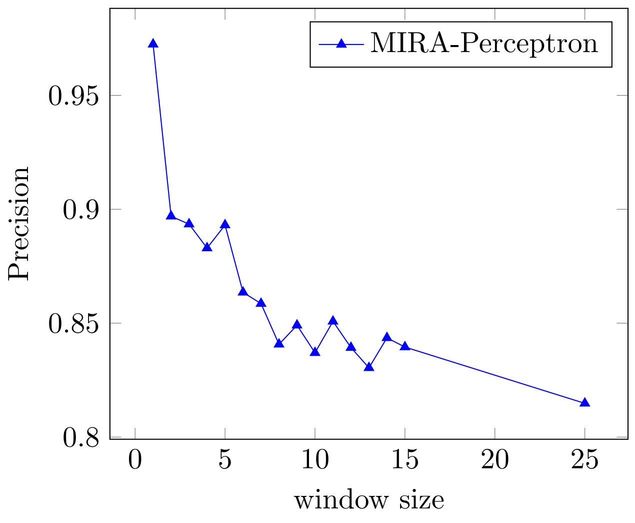


Figure 7.6.: Influence of the window size on the Precision during the 2-Way strategy on the development set. Increasing the window size to six reduces the Precision from 97 % down to about 86 %.

Table 7.5.: Amount of direct speech utterances in the different splits of the data, as well as the amount of utterances where a frame could be detected. In at least 2/3 of all cases, a frame could be detected.

Data split	**All utterances**	**Utterances with frame**	**Ratio**
Train	3200	2111	66 %
Validation	645	460	71 %
Test	732	484	66 %

Table 7.6.: Results of the individual approaches for the speaker detection of utterances with an assigned frame. Numbers are given in %.

Approach	Accuracy	Precision	Recall	F1
Rule-based	87.0	**94.2**	**93.6**	**93.9**
MaxEnt	88.2	92.0	90.1	91.0
Perceptron	88.0	91.6	89.9	90.7
LSTM	**89.0**	91.0	88.4	89.7
Transformer	87.4	89.8	86.7	88.2
Ranking	87.8	87.8	91.0	89.4

Table 7.7.: Results of the individual approaches for the speaker detection with candidates in according frames of direct speech utterances. Numbers are given in %.

Approach	Accuracy	Precision	Recall	F1
Rule-based	**93.3**	**94.0**	**93.3**	**93.7**
MaxEnt	91.6	93.2	91.6	92.4
Perceptron	90.6	92.7	90.6	91.7
LSTM	90.4	91.4	90.4	90.9
Transformer	88.6	89.7	88.5	89.1
Ranking	92.4	92.4	92.4	92.4

The results of this approach are depicted in table 7.6. Since the performance metrics are not straight forward to understand, we refer to section 7.6 for an in depth description of the involved metrics, however it should be noted that accuracy is the most meaningful one for this task, because the number does directly correspond to the amount of correct resolutions.

The LSTM shows the highest accuracy with about 89 % and the rule-based approach reaches almost 94 % F1 but only 87 % accuracy (this difference is due to the accuracy also respecting true negatives). Both feature based classifier outperform the LSTM on the F1 but lack accuracy. The ranking system and the transformer perform worst with neither of the meaningful metrics reach 90 %.

We experimented with the combination of our rule-based system and systems based on machine learning (by incorporating the matched rules as additional features for the systems) but did not find any beneficial results.

Since the input of the utterances still does not differ between real direct speech and other categories, there is still somewhat of a mix in this task (This has an impact when comparing the rule-based system to the machine learning systems, the rule-based system detects different categories and does no longer try to predict a speaker for them, however if the utterance is e.g. a *thought* there is usually still a character reference annotated in our data). In order to fully evaluate the different systems on their pure ability to detect the speaker inside frames of direct speeches, we evaluated this setting as well, these results can be found in table 7.7

Table 7.8.: Results of the individual approaches for the detection of the addressees of direct speech utterances. We separate the quality for the detection of addressees *inside* the utterance and those that are located *outside* the utterance (but inside the frame). Numbers are given in %.

Approach	P	R	F1	P_{in}	R_{in}	F_{in}	P_{out}	R_{out}	$F1_{out}$
Rule-based	69.8	53.5	60.7	69.8	**83.7**	76.1	0.0	0.0	0.0
MaxEnt	73.3	54.9	62.8	73.8	78.6	76.1	29.5	18.7	29.5
Perceptron	**74.4**	56.7	64.4	73.8	78.6	76.1	**72.7**	21.3	33.0
SVM	71.3	**68.4**	**69.8**	**75.0**	83.6	**79.1**	45.6	34.7	**39.4**

For the detection of the speaker of direct speech utterances, all systems performed much better than in the other scenarios. Currently the rule-based system outperforms all machine learning methods as well as the ranking method, on both metrics, accuracy and F1. The ranking system yielded the best accuracy and is on par with the Maximum Entropy classifier when comparing the F1. Even though the Perceptron showed the most robust results in the other settings, it only managed to get an accuracy of 90.6 %. The neural network based on LSTM's constantly outperformed the Transformer. The reported results for the deep learning approaches were run 11 times (with different initialisation), applied early stopping on the validation set and the averaged scores are reported. We expect the deep learning approaches to outperform the other feature based classifier on larger data sets, but at the current point we lack the data to back off this statement.

For the evaluation of addressees we refrained from modelling the explicit and the propagation step into a single prediction step and only tried to detect explicit addressees of direct speech utterances instead, which is the easiest setup possible for the detection of the addressee. Since there are many utterances without an explicit addressee (e.g. in the test set about 66 % of all utterances), we restricted the performance metric to Precision, Recall, and F1, so that not detecting any addressee does not get rewarded too much. A true positive can only be obtained by predicting the correct *reference* that is marked in the gold data (this differs to the evaluation of the explicit speaker, where **any** candidate referring to the correct entity was accepted). Among the 639 utterances in the test data, there are 140 (about 22 %) with an explicit addressee located inside the utterance and 75 (about 11 %) that are located in an nearby frame. Providing a baseline for this scenario is tricky, but even a smart one can not exceed 33 %. The results of the detection of the addressee is given in table 7.8

The overall best performing model is the support vector machine. We experimented with different kernels and varied the regularisation parameter C in an interval of 0.001 to 10.0 and found the linear kernel using $C = 0.75$ to yield the best results on the validation data. The Perceptron is again slightly better than the MaxEnt classifier with both yielding exactly the same results for the detection of the explicit addressees inside utterances. The rule-based approach shows the worst performance since at the current stage it can not detect addressees that are located in nearby frames (this is mainly due

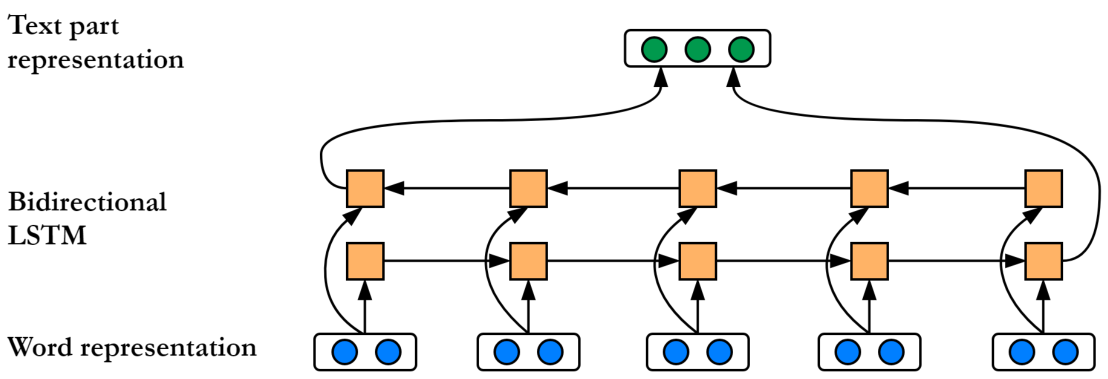


Figure 7.7.: The embedding of a text part using a BiLSTM layer.

to unreliable information given by the parser).

Comparing the best results for the detection of the explicit addressees (69.8 %) with the best results for the detection of the explicit speaker (about 93 %) shows, that the detection of the addressee is more challenging and amongst the detection of the addressees, the detection inside the utterance appears to be easier.

7.5. Parameters and model specifications for speaker detection

This section serves the purpose of elaborating the parameters as well as the ranges of parameters that were used during parameter search. The Maximum Entropy classifier only comes with a single parameter, namely the strength of the L2-regularisation. We varied it between 0.001 and 2.0 but found that a default value of 1.0 yields good results during all experiments. The Perceptron algorithm that is used uses the MIRA update [Crammer and Singer, 2003] to prevent overfitting and is ensembled after every parameter update by averaging the weights (see [Collins, 2002]). For our ranking Perceptron, we used the MIRA update (and weight averaging) paired with an Hinge-Loss during parameter update. In order to enable the ranking algorithm to predict no speaker for a given utterance, we added a dummy candidate into every ranking decision. The dummy candidate is enriched with a list of dummy features (namely all tokens inside the DS).

The neural networks embed a speaker by extracting 30 tokens to its left and to its right (padding is used if not enough tokens can be extracted) and map the tokens using a concatenation of pretrained Word2vec and GloVe embeddings. These sequences of tokens (called Text parts) are then embedded using either a BiLSTM layer (which map to 60 neurons in each direction, see figure 7.7) or a Transformer layer, which maps the sequence to an intermediate sequence that is concatenated and fed into a fully connected layer which again maps the representation to 120 dimensions (see figure 7.8). The Transformer uses a depth of three layers and a multi head attention of five [Vaswani et al., 2017]. After the text parts are embedded (there might be up to three text parts, 30 tokens before the speaker candidate, 30 tokens after the speaker candidate, and additional 30 tokens if there is more than one frame assigned to a direct speech utterance) they are concatenated

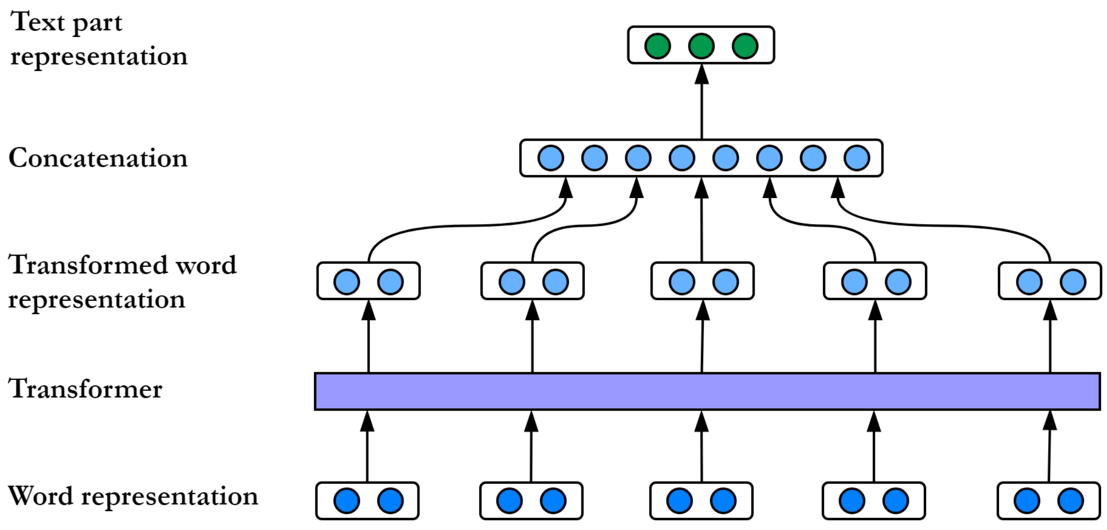


Figure 7.8.: The embedding of a text part using a Transformer layer.

with the embedded speech and two additional features that encode the position of the candidate in the frame (modelled as one-hot vectors). This architecture is depicted in figure 7.9. The networks are trained using early stopping on the development set with a batch size of 128, and a learning rate of 0.001 for the BiLSTM and 0.0001 for the Transformer respectively.

7.6. Performance metrics for speaker detection

The evaluation of the detection of the speaker (or addressee) is done as follows. A true positive is awarded, whenever any character reference of the correct entity was selected to be the speaker/addressee of an utterance (therefore we evaluate based on information given by gold coreference annotation). A false positive appears whenever a candidate was selected and turned out to be wrong, and a false negative appears whenever the system either suggests a wrong candidate and therefore failed to predict the correct speaker or the system did not predict any speaker when it should have. These statistics are collected and reported as Precision, Recall and F1-score. To be comparable to previous results, we also report the accuracy, that is the amount of correct resolutions (true positives + true negatives) divided by the total amount of direct speech utterances involved in an experiment. A true negative might occur if the annotator could not determine a suitable candidate herself. It is noteworthy that this evaluation scheme produces accuracy values that are lower than the corresponding F1-scores. Consider an algorithm, which is capable of detecting a speaker for 50% of all utterances, but only with a Precision of 90%. The resulting accuracy would be 45%, while the F1-score would be about 64%.

We used the same evaluation split as for the first fold of the experiments for coreference resolution, that is 72 documents of DROC are used as training data and 18 documents are used to test the algorithms. Since some approaches require the fine tuning of parameters

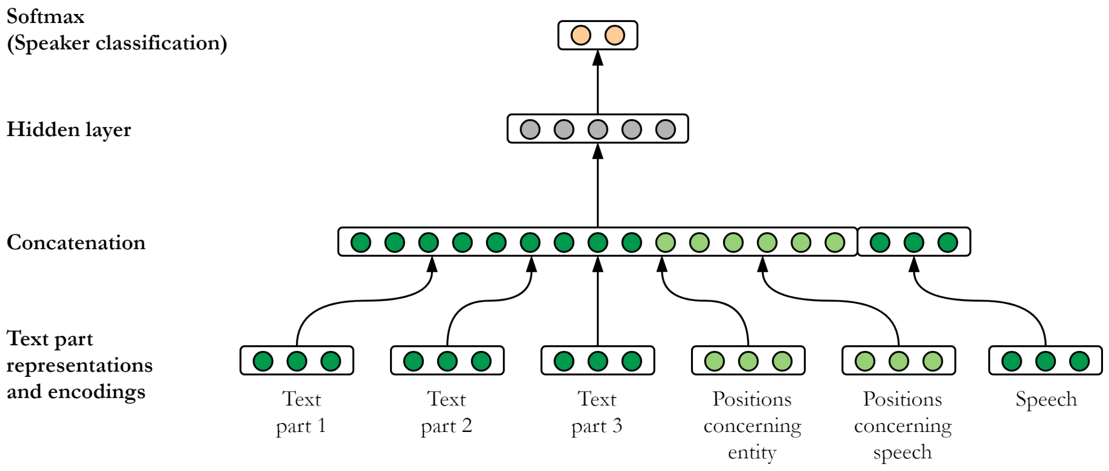


Figure 7.9.: The classification architecture of the network.

(such as the order in which we compare candidates, as well as the amount of candidates) we used 14 documents of the training data as our validation set.

7.7. Error Analysis of the Speaker Resolution

This section analysis the remaining mistakes of the rule based algorithm (since it is by far the best performing system, when considering the end-to-end approach). For this the mistakes (88 mistakes on 780 utterances) on the 18 documents of the test are analyzed and grouped into different categories. The largest category of mistakes is due to wrong propagations of speaker (43% of all mistakes). These errors are often times introduced by a different source of mistakes in advance (e.g. a wrong subject by the parser or a wrong addressee which was used for the propagation). The second source of mistakes arises from wrong parser information (33%), mainly due to the wrong detection of subjects. The assumption that whenever there is only a single candidate in the frame introduced 8% of the mistakes as well as a wrong coreference resolution, which caused the algorithm to believe that only two entities reside in a dialogue (8% of the mistakes). The heuristic, that whenever a colon could be detected prior to an utterance the closest candidate is the speaker is responsible for additional 5 mistakes (5.6%). The last two mistakes are due to a wrong combination of direct speeches that were assumed to be the same speech but ended up being different ones. Examples for the different categories are provided in figures 7.10, 7.11 and 7.12.

»[Herr Professor]«, erwiderte {mir} der Kapitän, »die Wälder, welche ich besitze, bedürfen weder Licht noch Wärme von der Sonne. Es hausen da weder Löwen, noch Tiger, noch Panther, oder sonst ein vierfüßiges Thier. Ich allein kenne sie. Es sind nicht Landforsten, sondern unterseeische.«[a]

– »Unterseeische Wälder!« rief ich aus.
– »Ja, Herr Professor.«
– »Und Sie wollen mich dahin führen?«
– »Ja wohl.«
– »Zu Fuß?« [b]

[a] »Professor," the captain replied to me, »the forests I own need neither light nor heat from the sun. Neither lions, nor tigers, nor panthers, nor a four-footed animal live there. I alone know them. They are not land forests, but undersea forests."

[b] - »Underwater woods!" I cried out.
- »Yes, professor.«
- »And you want to take me there?«
- »Yes, indeed.«
- »On foot?«

Figure 7.10.: An example of a wrong subject determination: The parser determined {mir} to be the subject which resides inside a frame and therefor a wrong speaker is detected. In the same utterance, **Herr Professor** is detected to be the addressee (which is correct) and the dialogue then continues and the propagation introduces another five mistakes. Even though **ich** could be detected as a new speaker, due to the coreference resolution which is applied in between it is resolved to **mir** and the mistake is retained. All DS are marked in bold. Example taken from Jules Verne - *Zwanzigtausend Meilen unter'm Meer*

Der Khabir sah uns mit scharfem, stechendem Blicke an und fragte dann {Kamil} in grollendem Tone:
»Dein Name ist Kamil Ben Sufakah? Zu welchem Volke gehörest du?«[a]

[a] The Khabir looked at us with sharp, piercing eyes and then asked {Kamil} in a rumbling tone: »Your name is Kamil Ben Sufakah? To what people do you belong?«

Figure 7.11.: An example of a wrong resolution due to the colon heuristics. Kamil was awarded with the score, but is not the correct speaker in this example. Example take from Karl May - *Auf fremden Pfaden*

Ein kleines Ponygefährt war schon vorher bis dicht an das Stationsgebäude herangefahren, und ein junges {Mädchen} von kaum sechzehn Jahren hielt die Zügel in Händen. »Grüß dich Gott, Ruth!«[a]

[a]A small pony car had already been driven close to the station building, and a young {girl} of barely sixteen years held the reins in her hands. »Greetings to you, Ruth.«

Figure 7.12.: An example for a wrong resolution to the only candidate in the context of an utterance. Example take from Theodor Fontane - *Quitt*

7.8. Conclusion

Speaker detection in literary fiction is not an easy task but is fruitful for a downstream coreference resolution. In this chapter, different techniques for the detection were examined. We found, that machine learning methods are on par with rule-based systems for the detection of explicit speaker and addressees, but whenever the machine learning algorithms are asked to incorporate the propagation step their quality drops significantly. In total we were able to produce a quality of about 85 % for the detection of the speaker and about 65 % for the detection of the addressee.

8. On the Adaption of Coreference Resolution to German Historic Novels

[1] Coreference Resolution is a crucial part for the analysis of global information prevalent in texts. Usually algorithms are designed and tailored towards newspaper articles, which tend to be rather short. This work examines the impact and quality of state of the art algorithms when the domain is changed to (German) literary fiction, especially novels. Not only are the documents much longer, often exceeding hundreds of pages, but new characteristics and the challenges of the domain are also introduced into the problem. Most notably the attribution of quotations, that is the detection of speaker and addressees, plays a crucial role for a coreference resolution algorithm that may still yield acceptable performance on entire novels. We compare the quality of previously released state of the art methods and show how to adapt them for the new domain to improve their quality. This chapter covers rule-based approaches, approaches based on traditional feature based machine learning as well as deep learning approaches for coreference resolution. It provides intrinsic results on the task at hand as well as end-to-end scores for coreference resolution for German literary texts. We show that rule-based systems can still perform on par with modern deep learning approaches while conserving explicability. Since none of the systems manages to overcome challenging semantics involved into coreference, we provide an error analysis for the best performing systems and give an outline as of which results of an algorithm can be trusted for subsequent analysis such as the extraction of character networks.

This section presents the coreference resolution systems that were examined in this work. The architectures of the aforementioned papers have been reimplemented and were evaluated on the DROC corpus. An important point to note is that our usage of the term mention differs from that in other works, like that by Lee et al. [Lee et al., 2013] for example. While Lee et al. call the whole expression *the beautiful mother* a mention, for us the mention is what Lee et al. call the head of the mention (*mother*). When we want to refer to the complete expression (*the beautiful mother*) we use the term chunk. Another point to note is that in German the text of a word or phrase depends on its case, e.g. for the nominative phrase 'die Männer' (the men) the dative version is 'den Männern'. We therefore usually also look at the lemmata of the words when we write that we compare the text of two mentions.

[1]The major part of this chapter is submitted for publication [Krug et al., 2019b]

8.1. Related Work

The following section introduces the most influential papers in the area of coreference resolution.

A lot of systems have been developed to try to solve the problem of coreference resolution. Like most algorithms for natural language processing, they can be divided into three groups: rule-based systems, systems based on classical machine learning and systems which use neural networks. In this section, we will cover the algorithms that have been used in this paper and some other influential works. A much more complete overview over systems for coreference resolution can be found in [Poesio et al., 2016], for example.

Rule-based systems Rule-based systems started with a greedy procedure, proposed by Hobbs [Hobbs, 1978]. His algorithm uses parse trees to check the potential antecedents of a pronoun in a specific order until the first candidate which satisfies all constraints is found. Sidner developed an algorithm called focussing [Sidner, 1979] for the resolution of personal pronouns and noun phrases used with a definite article, which picks an antecedent from a list of foci. She did not implement the algorithm herself, but the systems SPAR by Carter [Carter, 1986] and RAFT/RAPR by Suri and McCoy [Suri and McCoy, 1994] are based upon it. The algorithm of Brennan et al. [Brennan et al., 1987] is an implementation of the centering theory, which is based upon Sidner's focussing.

The next algorithm with high influence is the rule-based system of Lappin and Leass [Lappin and Leass, 1994], who treated the problem as a ranking problem between a candidate mention and previously introduced entities. More precisely they kept track of previously mentioned entities and scored the agreement of a candidate mention, with the highest scoring entity being used as the antecedent. Baldwin's system CogNIAC [Baldwin, 1996] consists of six high-precision rules which are applied to each pronoun. In contrast to most other algorithms before it provided the possibility to not resolve a pronoun at all (when none of the rules matched).

Similar to Lappin and Leass, Mitkov's algorithm [Mitkov, 1998] filters the possible antecedents and then scores them using antecedent indicators. One important difference is that Mitkov's algorithm only requires information from a part-of-speech tagger and a noun phrase extractor. The algorithm was later re-implemented and extended by Mitkov et al., resulting in the system MARS [Mitkov et al., 2002], which relied on information provided by a dependency parser for some of its new indicators and could detect expressions featuring *Expletive it* that cannot be resolved to an antecedent (because they refer to a sentence or are part of an expression like *It is important* for example).

The main difference of the rule-based algorithm developed by Raghunatan et al. [Raghunathan et al., 2010], and later improved by Lee et al. [Lee et al., 2011] and further extended by [Lee et al., 2013], to most other systems before that, is that it did not try to resolve all mentions in one pass over the text. Instead it splits the rules into several *sieves* which are applied to the text one after the other, so each sieve can build upon the results of the preceding sieves. The order of the sieves is determined by their precision:

more precise sieves are used first, while sieves with low precision like the one handling pronouns are applied at the end. In modern day language this can be considered as a version of *easy first classification*.

Machine learning models Among the first to use machine learning for coreference resolution were McCarthy and Lehnert [McCarthy and Lehnert, 1995] and Aone and Bennett [Aone and Bennett, 1995]. Both used the C4.5 decision tree algorithm to decide whether two mentions are coreferent and clustered the mentions based on these decisions. In both works, feature sets were tailored to their specific domain. Soon et al. [Soon et al., 2001] created a set of features for this mention-pair model which is independent of a specific domain.

One weakness of the mention-pair model is that the decision whether or not two mentions are coreferent is made with little information about the entities which the mentions represent. Because of that Yang et al. [Yang et al., 2004] extended the mention-pair model with features which did not only depend on the mention and a potential antecedent but also the other mentions in the antecedent's cluster, in this work referred to as the entity-mention model.

Another weakness of the mention-pair model is that each potential antecedent is classified separately, which means the model cannot tell whether a potential antecedent is better than another candidate. The system developed by Iida et al. [Iida et al., 2003] solved this by using a SVM classifier to decide which of two candidates is the better antecedent for a mention and ranking all potential antecedents with the help of this information (mention-ranking model). The idea to rank potential antecedents like this originates from the work of Connolly et al. [Connolly et al., 1997].

Rahman and Ng combined the improvements of the entity-mention model and the mention-ranking model in their system [Rahman and Ng, 2009]. This cluster-ranking model ranks the clusters of potential antecedents by using pairwise comparisons and picks the best cluster as antecedent. Fernandes et al. created an algorithm based on a latent tree model [Fernandes et al., 2014]: It treats coreference clusters as trees where each mention is a child node of its antecedent and connects the root nodes of all these trees to an artificial root node to create one tree for the complete clustering. A Structured Perceptron is used to learn to predict the best such tree for a given set of mentions. Additionally, they applied entropy-guided feature induction on hand-crafted features to create conjoined features for their algorithm. Björkelund and Kuhn extended this algorithm with the inclusion of non-local features and delayed learning as search optimisation [Björkelund and Kuhn, 2014]. Rösiger and Kuhn then adapted it to German texts [Rösiger and Kuhn, 2016].

Neural networks Wiseman et al. created a different kind of mention-ranking model [Wiseman et al., 2015]: Their system uses a neural network to compute a score for a mention and a potential antecedent and then picks the candidate with the highest score as antecedent. By adding a dummy mention to the list of potential antecedents of a mention, the system combines coreference resolution with anaphoricity detection (the

Table 8.1.: Results of various coreference resolution systems (Precision, Recall and F1 in %) on CoNLL-2011 (C11), CoNLL-2012 (C12) and ACE-2005 (ACE). The mention-pair, entity-mention, mention-ranking and cluster-ranking models are from [Rahman and Ng, 2009].

System	**data**	**MUC**			$\mathbf{B^3}$			$\mathbf{CEAF_E}$		
		P	R	F1	P	R	F1	P	R	F1
Lee et al. 2013	C11	62.1	65.9	63.9	70.6	69.5	70.0	50.5	46.3	48.3
Mention-pair	ACE	56.4	70.0	62.5	57.9	50.8	54.1	51.0	56.1	53.4
Entity-mention	ACE	57.2	68.5	62.3	57.8	51.2	54.3	50.2	56.3	53.1
Mention-ranking	ACE	73.0	62.1	67.1	65.5	50.4	56.9	58.5	53.0	55.6
Cluster-ranking	ACE	75.4	64.1	69.3	70.5	54.4	61.4	62.6	56.7	59.5
Fernandes et al.	C12	75.9	65.8	70.5	77.7	65.8	71.2	43.2	55.0	48.4
Wiseman et al.	C12	76.2	69.3	72.6	66.2	55.8	60.5	59.4	54.9	57.1
Clark & Manning	C12	79.2	70.4	74.6	69.9	58.0	63.4	63.5	55.5	59.2
Lee et al. 2017	C12	81.2	73.6	77.2	72.3	61.7	66.6	65.2	60.2	62.6

decision whether a mention refers to any entity that has been mentioned previously). The loss function penalises the different error types (antecedent found where there is none, no antecedent found where there is one, wrong antecedent picked) differently.

Clark and Manning implemented a mention-ranking model similar to that of Wiseman et al. [Clark and Manning, 2016b]. In addition to that, they created a cluster-ranking algorithm that is trained with a learning-to-search algorithm and uses the mention-ranking algorithm to prune candidate clusters. In [Clark and Manning, 2016a] the mention-ranking algorithm learned the order in which the mentions are resolved using reinforcement learning.

Lee et al. published an end-to-end neural network for coreference resolution that integrates both, mention detection as well as coreference resolution [Lee et al., 2017]. It does not use any hand-crafted features but relies instead solely on the words as input.

Metrics Unfortunately, there is no straightforward metric for coreference resolution. Instead several different metrics were proposed over time, each with advantages and disadvantages. In this work we use MUC [Vilain et al., 1995], the cluster based metric B-Cubed [Bagga and Baldwin, 1998], CEAF [Luo, 2005], BLANC [Recasens and Hovy, 2011] and the lately introduced metric LEA [Moosavi and Strube, 2016]. A detailed introduction of the metrics along their strengths and weaknesses is given in section 3.1.

A comparison of coreference resolution systems is often difficult, especially for early works, because the authors used different metrics and corpora for evaluation. In table 8.1 the evaluation results of several of the more recent systems are displayed. Among the machine learning models implemented by [Rahman and Ng, 2009], the cluster-ranking model achieved the best results on the ACE-2005 corpus. The best-performing system based on neural networks is the end to end neural network system by [Lee et al., 2017]. The latent tree model by [Fernandes et al., 2014] performed even better with respect to the B^3 metric, but worse with respect to MUC and $CEAF_E$.

8.2. Preprocessing

Since in this chapter, the data in focus originates the literary domain, the next section compares DROC with TueBa-D/Z [Telljohann et al., 2006] in terms of coreference statistics, followed by a quick recap of the preprocessing algorithms which were done prior to the experiments.

8.2.1. Differences between DROC and newspaper corpora for coreference resolution

In this section, the documents in DROC are compared against the documents in TueBa-D/Z in terms of relevant statistics for coreference. The analyzed metrics comprise: the amount of entities per document, the amount of references per sentence and the amount of references per entity. The results are shown in table 8.2. These metrics are straight forward to extract from DROC, however this is different in TueBa. For TueBa, only entities are take into account for this statistic if any mention was marked with the label "PER". This means, that entities which are referring to persons but were never named do not appear in these statistics. To accomodate for this, the same numbers are also reported for DROC separately and can be seen in table 8.2.

Table 8.2.: Differences between DROC and TueBa-D/Z on different features.

Feature	**TueBa-D/Z**	**DROC**	**DROC-Named**
Entities per document	2.43	58.8	11.4
Amount References per sentence	0.58	2.8	1.97
Amount References per entity	5.98	9.84	34.72
Proportion pronominal references	44.2%	65.4%	71.5%
Proportion noun phrase references	13.3%	23.0%	11.7%
Proportion proper nouns	42.5%	11.6%	16.8%

The amount of pronominal references in the newspaper articles is very similar to what was determined by [Kabadjov, 2007]. In general there are more pronouns (about 65 %) in DROC compared to the 44 % in TueBa. The characteristics for entities that appear with a name are very different to the statistics of all entities. It becomes apparent, that named entities are more central (in average 34 references compared to just about 10). The central characters are appear very often as pronouns and only rarely by the usage of noun phrases.

A second aspect, namely the proportion of references that fall onto the entities that appear most frequently was compared between the two domains and in shown in figure 8.1. In TueBa, the most prominent entity does already make up for about 2/3 of all references, which would justify the hypothesis, that a newspaper article is written to be

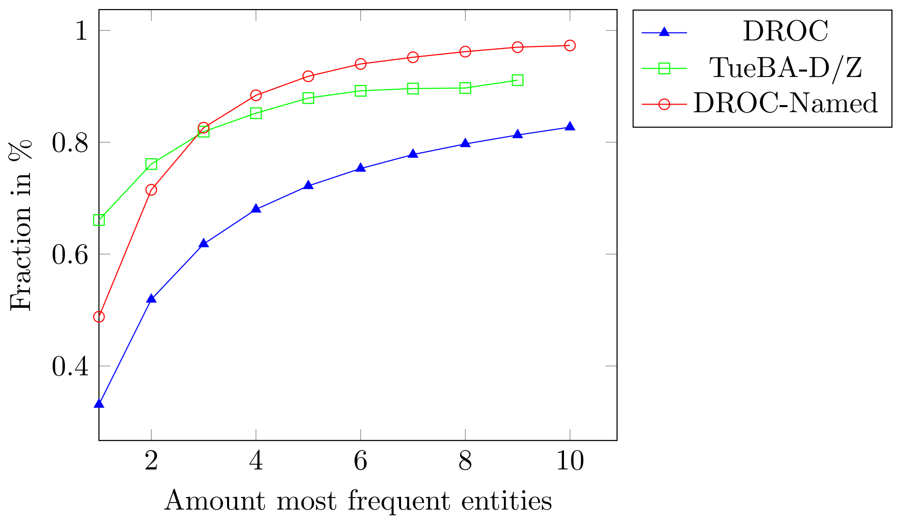


Figure 8.1.: The average proportion on references in a document that references the most prominent entities. For example the value at five shows the proportion of references of the top 5 entities in relation to all references that appear in the document.

about an entity. If only named entities are considered in DROC, then the four most central entities already make up about 90% of all references, as opposed to only about 60%, when all entities are considered.

8.2.2. Preprocessing of German novels

Since this work deals with rule-based, traditional machine learning as well as deep learning techniques, there is a need to preprocess the texts before applying both speaker detection as well as coreference resolution. This section gives a brief overview over the different components that are used to annotate the texts.

We split the document into sentences and tokens using the according German models of OpenNLP. [2] Part of speech tags, as well as a richer morphological analysis was annotated using the TreeTagger [Schmid, 1995] and the RFTagger [Schmid and Laws, 2008]. The information about chunks was added using the chunker that comes with the TreeTagger. Syntactical information in the form of a dependency tree was added using the Mate Toolkit [Bohnet, 2010]. The component presented in [Jannidis et al., 2017] was extended and combined with a rule-based approach and used to annotate mentions that serve as input for the coreference resolution. After the span detection of the mentions, we enrich them with more fine grained information. For each token of a mention we annotate, which semantic category it represents, this includes the distinction between *First Names, Last Names, Professions, Titles and Relation Indicator*. This is mainly done with the application of gazetteers as well as the application of special rule templates. This process

[2] `https://opennlp.apache.org/`, accessed 20.05.2020

is modelled as a pipeline using the Apache UIMA [Gotz and Suhre, 2004] framework.

The speaker detection requires additional preprocessing. The data used during the experiments is taken from project TextGrid. Most of these novels show reliable annotation of direct speech utterances using dedicated quotation marks. This is why we decided to just annotate direct speech instances based on simple regular expressions. This might not be sufficient in general, which means that more involved approaches would need to be used. For approaches towards automatic detection of direct speech utterances see e.g. [Scheible et al., 2016] or [Jannidis et al., 2018].

The input for the experiments based on deep learning is modelled as word embeddings. We pretrained GloVe embeddings [Pennington et al., 2014] as well as Word2Vec embeddings [Mikolov et al., 2013] on about 1.500 German novels from project Gutenberg.[3]

8.3. Methods

Rule-based approach The rule-based approach that was developed is an adaptation of Stanford's multi-pass sieve algorithm [Lee et al., 2013] that contains several new sieves. Most sieves iterate over the mentions in the order in which they appear in the text and try to find an antecedent for each. The potential antecedents are sorted similarly to Hobbs' algorithm [Hobbs, 1978]. Pronouns are handled separately from names and noun phrases: They are only resolved in the SpeakerResolutionSieve (first and second person pronouns in direct speech), PreciseConstructsSieve (relative and reflexive pronouns) and the two pronoun sieves (all other pronouns) and cannot be antecedents of names and noun phrases. This section presents the sieves we implemented in the order in which they are applied.

LinkMergingSieve When applying the algorithm to a document, the user can provide a list of links between mentions. Such a link can either indicate that two mentions belong to the same Entity (positive link) or that they do not and should therefore never be merged into the same cluster (negative link). This sieve merges the clusters of mentions for which there is a positive link, enabling the use of prior knowledge in the algorithm. Some sieves add negative links to this list to prevent merges in later sieves.

ExactStringMatchingSieve This sieve merges the clusters of all mentions which are names and have the same text (e.g. "Landsfeld" and "Landsfeld").

ChunkBlockingSieve A mention cannot be coreferent to another mention which appears in its chunk (e.g. *his father*), so for each such pair of mentions a negative link is created to prevent them from being merged into the same cluster in any of the later sieves.

SpeakerResolutionSieve In each direct speech utterance, the pronouns in first person singular are merged into one cluster together with the speaker and the pronouns in

[3]`www.gutenberg.org`, accessed 20.05.2020

second person singular are merged into another cluster together with the mention that is marked as addressee. Additionally, a negative link is created between the speaker and addressee clusters, so that the speaker and the addressee are guaranteed to end up in different cluster.

RelaxedStringMatchingSieve Mentions which are names or noun phrases are merged into one cluster if their chunks have the same text. This sieve prevents merges for family relation words but merges strings such as ”gardener” into a single cluster.

PreciseConstructsSieve Four types of constructions are resolved in this sieve:

- apposition (”Alexander, the boy, ...”)
- copula / predicate nominative (”He works as an architect”)
- reflexive pronouns
- relative pronouns

Most of the work is actually done beforehand. Appositions and copulae can be read from the tree the dependency parser generated. For the resolution of reflexive and relative pronouns, the tree is used, too. Reflexive pronouns usually refer to the subject of the clause and the predicate of a relative pronoun's clause should have a mention which is coreferent to the pronoun as its dependency head.

StrictHeadMatchingSieve In this sieve the cluster of a mention is merged into the cluster of a candidate if two conditions are satisfied:

1. The mention has the same text as a mention in the candidate's cluster.
2. All words in the mention's chunk except for definite articles and demonstrative pronouns appear in the chunks of the mentions in the candidate's cluster.

StrictHeadVarBSieve This sieve is a variation of the previous sieve. The first condition remains unchanged but the second condition is ignored.

RelaxedHeadMatchingSieve This is another relaxed version of the StrictHeadMatchingSieve. This time only the second condition is used.

HeadWordInclusionSieve The condition of the previous sieve is relaxed even further. Here, it is no longer necessary to find all words of the mention's chunk in the candidate's cluster, but only the words of the mention's text.

MetaDataNameSieve In a preprocessing step (described in section 8.2.2), an algorithm is used to determine the category of each word in a mention (e.g. first name, military title). These categories can be used to collect information about the entities represented by the clusters, like first name, last name, military or spiritual title or occupation. This information is stored in a meta data object, together with the entity's gender, and updated as clusters are merged. In this sieve, the clusters that have the same first name are merged into one cluster if all of them are compatible to each other, which means there may not be two clusters with conflicting information among them (e.g. different last names). The same is then done with clusters that have the same last name.

NicknameHeadMatchingSieve This sieve is only applied to mentions that consist of one word. Their word endings (e.g. diminutiv) are removed and a stemmer is used. If the resulting stems are equal for two mentions their clusters are merged.

NameMergeProhibitionSieve The purpose of this sieve is not the merging of clusters, but to prevent errors in later sieves. To that end, negative links are created for all names that appear in the same sentence and have not been merged into one entity already (e.g. "Harry and Ron are working together").

ChunkAttributesMatchingSieve In German, it is possible to nominalise an adjective in a noun phrase, like *alte* in *die alte Frau* (the old woman), and omit the original noun so the phrase becomes *die Alte*, which is a reference to the same entity. This sieve merges the clusters of mentions that fit this pattern.

PronounSieve and PronounDsSieve These sieves resolve all pronouns which were not resolved in one of the sieves before. Pronouns inside direct speech are handled separately from pronouns outside of direct speech, so they are resolved to a mention that is also inside or outside of direct speech.

FamilyRelationSieve This last sieve is intended to exploit specific relations between persons which are unambiguous in one direction. At the moment, the only relations types implemented are the mother and father relations (hence FamilyRelation-Sieve). If two relations of the same type are found in the text and the mentions on the ambiguous side are known to be coreferent, it can be concluded that the mentions on the unambiguous side are coreferent as well. Example: In one part of the text there is the phrase *Effi's mother* and in another part the mention *Mum* appears in a direct speech which Effi is the speaker of. Since the algorithm already determined that *Effi* from the first phrase and the speaker of the direct speech are coreferent, the mentions *mother* and *Mum* are coreferent as well.

Before applying the sieves, each mention is put into a separate cluster. When a sieve considers two mentions to be coreferent their clusters are merged. This allows each sieve to build upon its own decisions as well as the decisions of previous sieves. The clusters produced by the last sieve are the result of the algorithm.

Classical machine learning We implemented four coreference resolution models that are based on machine-learning (for a more detailed explanation see [Rahman and Ng, 2009]):

Mention-pair model This model iterates over the mentions that appear before it in the text and uses a classifier to decide whether two mentions are coreferent and builds the clusters based on these decisions.

Entity-mention model This model considers the clusters of potential antecedents when deciding whether or not it is coreferent to the mention which is to be resolved. We implemented this model in two variants: The first variant picks the first potential antecedent that is classified as coreferent (first link), the second has all potential antecedents classified and picks the one which has been classified as coreferent with the highest confidence (best link).

Mention-ranking model This model ranks the potential antecedents by using a classifier to decide which of two mentions is the better antecedent for the mention which is to be resolved. The candidate which wins the comparison is then compared to the next candidate and so on, until the last candidate has been compared to the previous winner (tournament model [Iida et al., 2003]). To allow the system to not resolve a mention at all (because some mentions are non-anaphoric and form singleton clusters) a null mention is included in the ranking.

Cluster-ranking model This model builds a ranking like the mention-ranking model while using features which include information about the potential antecedents' clusters like the entity-mention model.

Each of the models needs to be combined with a classifier and a set of feature generators. In addition to that, a sampling method can be used for training. The feature generators are used to extract the important information about the given mentions or clusters and their relationships. For this work, we compared three collections of feature generators:

- Soon feature collection (S): based on the feature generators by [Soon et al., 2001].
- Rahman feature collection (R): based on the feature generators by Rahman and Ng [Rahman and Ng, 2009]
- Wue feature collection (W): builds upon the Rahman Feature Collection (the additional features are described in the following paragraph)

Not all of the features mentioned in [Soon et al., 2001] and [Rahman and Ng, 2009] were implemented. Semantic class and animancy are irrelevant for our task, for example, since all the mentions that are handled are literary characters anyway. In addition to the feature generators of the Rahman Feature Collection the Wue Feature Collection consists of the following feature generators:

Reflexive head If one of the mentions is a reflexive pronoun the dependency parse tree is used to check whether the other mention is the subject of the verb that the reflexive pronoun has as its head.

Relative head This feature generator uses the dependency parse tree, too. It checks whether one of the mentions is a relative pronoun and should be resolved to the other mention (the relative pronoun has the clause's predicate as its ancestor; if the other mention is the first ancestor of this predicate which is located before the relative clause, the two mentions are coreferent).

Act in DS If the mention to be resolved is in a direct speech this feature generator returns true, otherwise it returns false.

Candidate in DS Does the same as the previous feature generator, but for the candidate mention.

Direct speech This feature generator can return three different values. The first one indicates that there is a direct speech where one mention is a first person pronoun and the other the speaker of the direct speech or one mention is a second person pronoun and the other is the person spoken to. The second value is returned if there is instead a direct speech in which the mentions cannot be coreferent (first person pronoun and spoken to or second person pronoun and speaker). If neither of those cases applies the feature generator produces a third (neutral) value.

Direct speech distance This is computed by counting the number of direct speeches between the two mentions. The result can either be odd, even, that they are both in the same direct speech or that at least one of them is not part of a direct speech.

Hobbs next The value returned by this feature generator depends on whether one mention is the first mention Hobbs' search would consider when searching for an antecedent for the other mention.

First name This feature generator checks the first names of the mentions. They can be the same, different or compatible (meaning that the first name of at least one of them is unknown).

Last name Works similar to the previous feature generator by regarding last names instead of first names.

Meta data This feature generator expands upon the two previous feature generators by not only checking the mentions for compatibility in first name and last name, but in titles, profession and gender as well.

The instances created by the feature generators (one for each pair or triple of mentions/clusters) are then given to the classifier to decide whether the two mentions/clusters are coreferent (in a classification setting) or which one is the better antecedent (in a ranking setting). During the training phase the instances also contain the gold label. The Mallet implementations [McCallum, 2002] of the following classifiers can be used:

- Decision-Tree classifier (DT)
- Maximum entropy classifier (ME)
- Naive Bayes classifier (NB)
- SVM classifier (SVM)

Table 8.3.: Baseline results on DROC (coref F1, noncoref F1 and general F1 for BLANC; Precision, Recall and F1 for other metrics, in per cent)

Metric	**One cluster**			**Singleton clusters**		
	P	R	F1	P	R	F1
MUC6	89.4	100.0	94.3	0.0	0.1	0.0
B^3	18.4	100.0	30.4	100.0	10.8	19.2
BLANC	30.0	0.0	15.0	0.0	89.8	44.9
$CEAF_E$	48.9	1.1	2.2	7.3	66.0	12.9
$CEAF_M$	33.1	33.1	33.1	40.3	40.3	40.3
LEA	18.1	95.1	29.8	5.0	4.9	5.0

An imbalanced set of training instances, where one label occurs much more often than the other(s), can lead to problems for machine learning algorithms [Chawla et al., 2004]. In the case of coreference resolution the training instances are usually highly imbalanced: For example, only about 5% of the instances created by the mention-pair model on DROC are coreferent. We have therefore implemented the following sampling methods for our experiments:

Up sampling (U) We take all instances that are not coreferent and randomly pick coreferent instances until there are the same number of instances of both types.

Down sampling (D) In this case the noncoreferent instances are picked at random until there are as many picked noncoreferent instances as there are coreferent instances.

Noop sampling (NO) This means that there is no sampling at all. All coreferent and all noncoreferent instances are used for training.

8.3.1. Evaluation

We evaluated our rule-based algorithm, the machine-learning models and the end-to-end system by Lee et al. [Lee et al., 2017] on the DROC corpus using the metrics MUC, B-Cubed, BLANC, CEAF and LEA. For an explanation of LEA and a short overview over the other metrics see [Moosavi and Strube, 2016], for example.

There are two baselines for the coreference resolution: Merging all mentions into one cluster and putting each mention in a separate cluster (singleton clusters). The results of these baselines can be seen in table 8.3.

Rule-based approach The results of the evaluation of our adaptation of the Stanford sieve is displayed in table 8.4. The algorithm reaches F1 values of 83.6 % (MUC), 54.8 % (B-Cubed), 70.2 % (BLANC), 42.4 % ($CEAF_E$) and 47.3 % (LEA). Except for $CEAF_E$ the precision is significantly better than the recall.

To examine the influence of the documents' size, we split the documents of DROC into chapters to evaluate on them and and we evaluated the results of the algorithm

Table 8.4.: Results of the adapted Stanford sieve algorithm on the DROC corpus, the chapters of DROC and the novel Effi Briest (Precision, Recall and F1, in per cent)

	MUC6			**B^3**			**BLANC**	**$CEAF_E$**		
	P	R	F1	P	R	F1	F1	P	R	F1
DROC	87.9	79.9	83.6	73.1	44.9	54.8	70.2	34.1	59.8	42.4
Chapters	84.1	75.2	79.2	78.4	52.1	61.4	70.7	40.1	66.3	48.3
Effi	90.0	81.9	85.8	52.8	31.6	39.6	-	21.2	50.8	29.9

on a complete novel (Effi Briest by Fontane). As expected, the results improve on the smaller documents and are worse on the complete novel (with the exception of the MUC metric).

Classical machine learning For the evaluation of the classical machine learning models we split DROC into five folds. The presented results are the averages over these five folds. Each model was evaluated with all combinations of the previously presented feature collections, classifiers (except SVM) and sampling methods. We omit the tables which list all results for each model and only describe them in the text. The best results of each combination of model and feature collection can be seen in table 8.5. While the mention-pair model performed statistically significantly better using our feature collection (Wue) than with the other feature collections, other models did the opposite (especially the cluster-ranking model).

With the mention-pair model using no sampling instead of up or down sampling often lead to better results on the BLANC and CEAF metrics, but worse results on MUC, B-Cubed and LEA. The differences between up and down sampling are usually less than half a %. The best results on all metrics but MUC and $CEAF_E$ were obtained with the combination of Wue feature collection, Maximum Entropy classifier and no sampling: 81.5 % (MUC), 44.7 % (B-Cubed), 61.3 % (BLANC), 41.7 % ($CEAF_E$), 42.8 % ($CEAF_M$), 37.4 % (LEA).

For the entity-mention model (first link), the combination of Rahman feature collection, decision tree classifier and noop sampling delivered the best results with respect to all metrics but MUC: 78.5 % (MUC), 40.6 % (B-Cubed), 60.9 % (BLANC), 41.9 % ($CEAF_E$), 38.7 % ($CEAF_M$) and 32.4 % (LEA).

In contrast to the other two models the best link version of the entity-mention model has no combination that obtained the best results with respect to the majority of metrics. The Rahman feature collection together with the decision tree classifier and no sampling has the best results with respect to BLANC and $CEAF_E$ while the combination of Rahman feature collection, Maximum Entropy classifier and upsampling has the best results with respect to $CEAF_M$ and LEA. The latter is overall better than the former since its results are better with respect to MUC and B^3 as well. It achieved F1 values of 92.0 % (MUC), 36.1 % (B-Cubed), 34.1 % (BLANC), 12.3 % ($CEAF_E$), 35.8 % ($CEAF_M$) and 33.6 % (LEA). The results of the Naive Bayes classifier together with the Rahman

Table 8.5.: Results of the mention-pair (MP), entity-mention (EMf/EMb), mention-ranking (MR) and cluster-ranking (CR) models on DROC (F1 in per cent). F: feature collection (Rahman, Soon, Wue), C: classifier (decision tree, maximum-entropy, naive-bayes), S: sampling method (down, noop, up)

	F	**C**	**S**	**MUC**	**B^3**	**BLANC**	**$CEAF_E$**	**$CEAF_M$**	**LEA**
MP	R	ME	NO	80.3	43.2	60.4	39.9	41.2	35.8
MP	S	NB	D	86.9	36.4	51.1	21.7	32.3	31.9
MP	W	ME	NO	81.5	44.7	**61.3**	41.7	42.8	37.4
EMf	R	DT	NO	78.5	40.6	60.9	41.9	38.7	32.4
EMf	S	ME	NO	75.4	38.3	56.7	36.9	37.3	29.6
EMf	W	DT	NO	77.6	39.4	59.8	40.6	37.4	31.0
EMb	R	ME	U	92.0	36.1	34.1	12.3	35.8	33.6
EMb	S	ME	U	**94.0**	31.8	20.2	4.1	33.8	31.0
EMb	W	DT	U	92.1	33.4	36.6	7.4	34.1	31.8
MR	R	ME	NO	67.2	32.7	52.4	29.9	30.8	23.5
MR	S	ME	NO	68.5	33.6	53.4	30.4	31.6	24.3
MR	W	ME	D	69.0	32.8	51.5	25.5	27.8	22.4
CR	R	DT	NO	81.0	41.3	58.2	42.4	39.1	33.8
CR	S	ME	NO	85.6	**46.5**	58.2	**42.8**	**44.0**	**39.8**
CR	W	ME	NO	80.2	45.4	54.7	36.7	40.3	36.8

and Wue feature collections are notable: In all six experiments the algorithm did not cluster any of the mentions together but instead left each of them in a separate cluster. The combination of Wue feature collection with the Maximum Entropy classifier and up or down sampling came very close to the other baseline (all mentions in one cluster).

As was the case with the entity-mention model, the mention-ranking model leaves all mentions in separate clusters when used with the Naive Bayes classifier, only this time in combination with all sets of feature generators. The best results with respect to all metrics but MUC were obtained with the Soon feature collection, the Maximum Entropy classifier and noop sampling: 68.5 % (MUC), 33.6 % (B-Cubed), 53.4 % (BLANC), 30.4 % ($CEAF_E$), 31.6 % ($CEAF_M$) and 24.3 % (LEA).

The cluster-ranking model has the best results with respect to B^3, $CEAF_E$, $CEAF_M$ and LEA when used with the Soon feature collection, the Maximum Entropy classifier and no sampling. The F1 values are 85.6 % (MUC), 46.5 % (B-Cubed), 58.2 % (BLANC), 42.8 % ($CEAF_E$), 44.0 % ($CEAF_M$) and 39.8 % (LEA). Combining the Naive Bayes classifier with the Soon feature collection leads to all mentions being put into one cluster. Changing the feature collection improves the results only slightly.

We did not include any results with the SVM classifier since it was very slow and we could only run very few experiments because of that. From the experiments we did run with the SVM classifier it appeared that one of the other classifiers was better in most cases. Only the entity-mention model (first link) had slightly better results with the SVM classifier than with the other classifiers.

Table 8.6.: Results of the implemented algorithms on DROC (F1 in per cent)

Model	**MUC**	**B^3**	**BLANC**	**CEAF$_E$**	**CEAF$_M$**	**LEA**
Stanford sieve	83.6	**54.8**	**70.2**	42.4	**53.8**	**47.3**
Mention-pair	81.5	44.7	61.3	41.7	42.8	37.4
Entity-mention (first-link)	78.5	40.6	60.9	41.9	38.7	32.4
Entity-mention (best-link)	**92.0**	36.1	34.1	12.3	35.8	33.6
Mention-ranking	68.5	33.6	53.4	30.4	31.6	24.3
Cluster-ranking	85.6	46.5	58.2	**42.8**	44.0	39.8

Somewhat surprisingly, all models except for the best-link version of the entity-mention model had the best results without sampling. Three out of four times this was in combination with the Maximum Entropy classifier.

Table 8.6 shows the best results of all the models we implemented. The cluster-ranking model is the best machine learning model according to four of the six metrics, followed by the mention-pair model. The mention-ranking model is the worst model with respect to MUC, B-Cubed, CEAF$_M$ and LEA. Our adaptation of Stanford's sieve algorithm is better than all machine learning models according to four metrics.

Neural networks We evaluated the system of [Lee et al., 2017] on the first fold of DROC after about 380,000 steps of training (we only let the system consider mentions which had a size of up to four words). In addition to the results of the coreference resolution we also evaluated the mentions found by the neural network. The results of the named entity recognition were 93.2 % precision, 82.3 % recall and 87.1 % F1. False positives were mostly caused by pronouns while the majority of false negatives consisted of names and noun phrases.

The results of the neural network on the first fold can be seen in table 8.7, together with the results of the rule-based approach and the classical machine learning models on this fold. The neural network achieved F1 scores of 87.5 % (MUC), 40.4 % (B-Cubed), 49.9 % (BLANC), 31.6 % (CEAF$_E$) and 36.9 % (LEA). With the exception of the MUC metric, the sieves algorithm was significantly better with respect to all metrics. The cluster-ranking model and the mention-pair model were better than the neural network according to at least three of the metrics. An important point to consider here is that the neural network did not use gold mentions, in contrast to the other systems. The next sections compare the end to end neural network with our end to end pipeline approach, followed by an error analysis in two documents of the test set.

8.3.2. Influence of document size on the results

Evaluating the results of automatic coreference is problematic (otherwise there would not be a need for so many different evaluation metrics). In this section, we examine the characteristics of the metrics. In order to do this, we applied our rule-based algorithm to all documents of DROC and the approach based on deep learning to the test documents of

Table 8.7.: Results of the End-to-end neural network, the sieves algorithm and the classical machine learning models on one fold of DROC (Precision, Recall and F1 in per cent).

	MUC6			**B^3**			**BLANC**	**$CEAF_E$**		
	P	R	F1	P	R	F1	F1	P	R	F1
E2E-NN	89.5	85.7	87.5	64.9	30.2	40.4	49.9	32.9	31.3	31.6
Sieve	86.1	78.2	81.9	71.4	44.3	**53.9**	**69.5**	33.7	59.6	**41.8**
MP	89,0	72,6	79,8	76,0	31,2	43,5	61,0	28,6	70,6	40,1
EM fl	80,4	70,8	75,2	64,0	30,5	40,2	60,9	31,0	61,4	40,3
EM bl	90,4	97,8	**93,9**	24,0	85,4	36,4	33,5	39,1	5,6	9,4
MR	80,0	55,7	65,5	75,5	21,8	33,1	53,5	19,7	66,8	29,7
CR	86,6	82,5	84,5	46,2	47,0	46,1	58,1	36,8	50,3	**41,8**

the first fold (the same documents on which we evaluated the machine learning methods and the end to end algorithm) and varied the evaluation scenario. For this, we split the documents into smaller pieces, on the one extreme, we split all documents into pieces of the length of one single sentence up to a document length of 100 sentences, the results are shown in figure 8.2. This not only gives insight about the stability of the metrics towards document length, it also allows to compare the metrics with each other. Comparing the results, MUC appears to be the most stable metric when increasing the document sizes. Interestingly, all four cluster based metrics show exactly the same trends with B-Cubed and $CEAF_M$ reporting almost the same values. Except for MUC all metrics tend to drop with increasing size of the documents with $CEAF_E$ being the most stringent of those metrics. The plots suggest that the cluster metrics are almost redundant when reporting evaluation results, when ignoring a small bias among them. In any case, there should be at least one metric that is based on edges (that is either MUC of BLANC) and at least one metric that evaluated the clustering itself (CEAF, B-Cubed or LEA).

8.4. End-to-end coreference resolution

In the last section, we described the coreference resolution algorithms we implemented and their evaluation on DROC. The information about character references, direct speeches, speakers and addressee that was used in this evaluation was gold data. In this section, we evaluated the performance of the rule-based algorithm and the best classical machine-learning model (cluster-ranking) when they are provided only with system information.

Table 8.8 shows the results of the sieve algorithm and the cluster-ranking model on the first fold on DROC when only system information is provided. For comparison, Lee et al.'s neural network is listed as well. The sieve algorithm is the best-performing system with respect to B^3 (41.0 % F1), BLANC (53.4 % F1) and $CEAF_E$ (32.7 % F1), but the worst with respect to MUC (80.9 % F1). The neural network is the best system according to MUC (87.5 % F1) and the worst according to $CEAF_E$ (31.6 % F1). It lies between

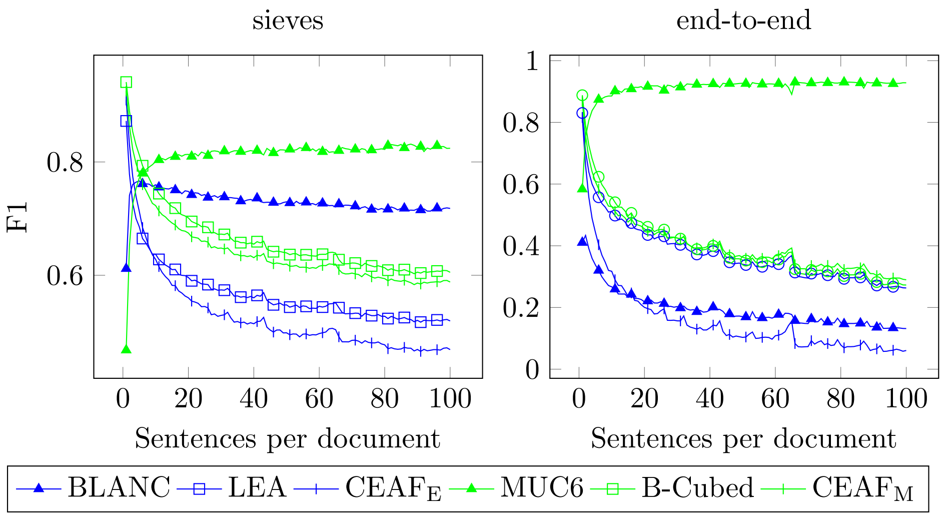


Figure 8.2.: Results of the sieves algorithm (left) and the end-to-end algorithm (right) on split documents of various sizes

Table 8.8.: Results of the sieves algorithm, the cluster-ranking model and Lee et al.'s neural network on the first fold of DROC using system mentions.

	MUC6			**B^3**			**BLANC**	**$CEAF_E$**		
	P	R	F1	P	R	F1	F1	P	R	F1
Sieve	83.1	78.9	80.9	52.2	35.4	41.0	53.4	27.9	42.3	32.7
CR	86.4	82.7	84.5	34.7	42.7	37.1	44.5	30.0	36.9	32.4
E2E-NN	89.5	85.7	87.5	64.9	30.2	40.4	49.9	32.9	31.3	31.6

the other two models with respect to B^3 (40.4 % F1) and BLANC (49.9 % F1). The cluster-ranking model performed either worse than one of the other models (MUC and $CEAF_E$) or both (B^3 and BLANC). While the comparison in table 8.7 disadvantaged the neural network, it has an advantage in this comparison: It could use gold information about direct speeches and speakers.

A more fine-grained analysis about the individual errors that are made by the systems is provided in the next section.

8.5. Error analysis

To be able to get an idea of the problems of the current approaches we conducted an error analysis by inspecting two different documents of DROC and the results of two different algorithms. In particular, we compared the results of the end-to-end neural network with our rule-based approach on *Karl May- Auf fremden Pfaden* and on *Louise Aston - Lydia*. The results of these two algorithms on the selected documents are shown in table 8.9.

Table 8.9.: Results of the sieves algorithm and the neural network on the documents *Lydia* by Louise Aston and *Auf fremden Pfaden* by Karl May (F1 in %).

Document	algorithm	MUC	B^3	BLANC	$CEAF_E$	$CEAF_M$	LEA
Aston	sieve	84.9	45.2	63.1	25.9	45.3	40.6
May	sieve	78.9	31.9	45.6	19.4	33.1	26.3
Aston	nn	89.8	32.7	50.6	24.0	30.3	30.7
May	nn	88.1	38.8	50.1	28.6	36.8	35.9

The approach based on neural networks performs much worse on *Lydia* compared to the rule-based approach, but outperforms it by a large margin for *Auf fremden Pfaden*. We analysed the most dominant sources of mistakes for both systems in order to explain these differences.

End-to-end neural network: In *Lydia*, the network starts out flawlessly by recognizing the two main characters *Lydia* and *Landsfeld*, even though Landsfeld is also referred to as *Geliebter, ("beloved")*. The problems start in the next section, where the mother of Lydia is mentioned and a dialogue between Lydia and Landsfeld forms the center of this section. The algorithm decided, that all three entities are one and the same and therefore introduces many mistakes at once. This property holds throughout the entire document. After a couple of sentences the algorithm tends to forget what happened before and clusters many of the main characters into a single entity. This is repeated periodically till the end of the document.

In the second document - *Auf fremden Pfaden*, the algorithm tends to remember the entities for a longer period. The segments deal with a conversation between three male characters, and during the first half of the document the algorithm can differentiate between the entities. After about half of the document, we were not able to determine any sort of logical correlations of the results and the algorithm mixes the entities till the end of the section is reached.

Rule-based pipeline: The evaluation of the rule-based pipelines on those two documents revealed more understandable results. For *Lydia*, the following categories of mistakes could be detected:

- Correct chains of pronouns are resolved to a wrong reference due to a) the correct reference was not detected by the mention detection or b) wrong gender or number caused the algorithm to discard the correct reference. (E.g the mention detection algorithm marked the token *Aether ("ether")* as a reference. An example for a wrong morphological attribute is that *Landsfeld* is treated as a mention with unknown gender instead of male.)

- A wrong speaker/addressee propagation was the reason for the algorithm to introduce new entities instead of reusing existing ones.

- Especially during dialogues, entities are referred to with different texts, e.g. *Richard* is referred to as *Gemahl ("husband")* or *Geliebter ("beloved").*
- Situations in which semantics or more background information about the characters is necessary. One such instance is that *Gertrud* is the servant of *Lydia* or *Carl* is the servant of *Landsfeld.*

Even though the mention detection stage itself showed a good quality, the mistakes that are made may still result in serious problems for a following coreference system, especially if they mix with other sources of errors.

The second document - *Auf fremden Pfaden*, in which the rule-based approach only reached a B^3-F1 of about 32 % shows a completely different pattern. The document deals with mainly arabic characters which tend to have really long names (e.g. "Hadschi Kara Ben Nemsi") and at least for the western person rather unknown titles (such as "Sihdi"). This is problematic since *Sihdi* is one of the main characters of this section and no chain can therefore be resolved to it. Secondly, the arabic names are split into distinct references, which means that e.g. *Hadschi Kara Ben Nemsi* gets split into four different mentions and subsequently creates more anchors for a pronominal resolution. Just considering the automatic reference detection reveals that the quality of this documents changes from about 45 % B^3-F1 down to just 32 % revealing a severe impact of a high quality mention detection.

Summarising the error analysis reveals that the rule-based algorithm can especially be improved by extending the character reference detection to different regional naming conventions. Another improvement might result from focussing even stronger on the dialogues itself. By combining entities that are introduced in a paragraph or sentence preceding a dialogue or in the frame of a direct speech utterance with those entities that are detected inside the utterances. This could yield large improvements, e.g. in *Lydia* this could infer that *Landsfeld* is called *Richard* and is the husband and beloved of *Lydia.* This in turn could infer the gender of *Landsfeld* and prevent a wrong resolution of pronominal chains. The neural network shows some incredible results in some local segments of the text but fails completely in some other passages by mixing all entities into a single one. Not only that but applying the current algorithm to an entire novel requires more than 500GB of memory. Since it is very hard to even follow any sort of logical behaviour of the algorithm when inspecting its results, we propose to use it on short local sections and combine the results with a rule-based algorithm which enforces constraints as good as possible.

8.6. Usage of the Intuitive Evaluation Metric

During this thesis (see section 3.1.3), a new metric for coreference resolution was proposed and can now be used to the results of the rule based pipeline[4]. For this experiment, for

[4]Its detailed output can only be applied to the rule based pipeline, because the information about the type of character reference is available. In general the metric can be applied to all results but the less information is available, the less information can be used for the explainability of the mistakes.

Table 8.10.: Results of the fine-grained metric for coreference evaluation. The columns represent the average numbers through all 18 documents.

Evaluation-Type	Precision	Recall	F1
Merge-Entity	58.1	54.7	56.2
New-Entity	35.1	48.5	39.7
Merge-Entity-Appellative	43.9	20.2	27.2
New-Entity-Appellative	44.1	46.1	42.9
Merge-Entity-Core	61.0	62.2	60.1
New-Entity-Core	39.7	62.0	47.4
Merge-Entity-Pronoun	58.0	62.3	59.8
New-Entity-Pronoun	18.1	46.8	24.5

the 18 documents of the test data. The system pipeline is applied and this time it is evaluated using the proposed metric. See table 8.10

At the top-level, this evaluation shows, that the algorithm is better at merging, than at the detection of when a new entity is introduced in the text. The merging of the names works best, but does still only reach F1-scores of about 60% (this metric can still not differentiate between mistakes originating wrong merges and mistakes resulting from the mention detection). Appellatives represent by far the worst category and it becomes apparent, that the rule based pipeline is not capable of merging them with a good quality. The quality of pronoun resolution is about 60% for these documents, both when it comes to the Precision and the Recall.

If one would average both top-level F1-scores, then the result would also be situated below 50% and the statement of the metric would be comparable to that of the other metrics. In total this analysis offers useful information about the results of the algorithm that can be interpreted by a human and offer more insight into the core problems of an algorithm.

8.7. Conclusion

In this work, we showed different approaches to adapt several algorithms for coreference resolution to German historic novels. For this we focused on just the references to characters. In section 8.2.1, we reported that one key difference to newspaper articles is that novels not only have much longer content but also a much higher proportion of pronominal references (about 65% vs about 44% in newspaper articles).

We deducted an extensive study on the application and adaption of different coreference algorithms based on classical machine learning, based on deep learning and based on the Stanford sieve algorithm. The results show that among classical machine learning approaches the cluster ranking showed the best results. And the Neural network yielded the overall highest MUC score. In all other metrics, the adapted Stanford sieve (which makes use of the rule-based speaker and addressee detection module presented in this work) showed the highest results.

We compared the results of three end-to-end systems, the first being our adapted pipeline, the second being the neural end-to-end system, and the third being the cluster ranking model, on 18 documents of DROC. The results of the neural network almost matched the results of the pipeline, but in general, all systems perform rather poor with just about 40% B^3-F1 score. In order to gain insights about future directions for further improvements we inspected the results of those two systems on two documents and found that those algorithms yield very different results with the neural network performing much better in a restricted context while the rule-based approach can respect global consistencies much better. We leave experiments for the combination of those two approaches for future work.

9. On the Extraction of Relations between Character References of German Historic Novels

[1] This chapter describes the current state of the relation detection as it is integrated into the Kallimachos Pipeline. The term relation detection is used ambiguously in the literature. For this chapter, it means the detection of a relation between two character references as well as a subsequent assignment of a label (this step is sometimes split up and referred to as relation classification). Relation detection is a rather complicated task, starting with the issues to obtain manually labelled data of high quality.
Being able to (reliably) extract relations between characters allows for a multitude of constraints in downstream tasks, such as:

- Providing background knowledge for a coreference module (e.g. if the term *mother* appears, it can only be resolved in the document if the information whose mother it is is available)
- It allows for a typed aggregation of interaction statistics throughout a text to be evaluated and displayed in a character network (see section 9.4)
- It allows for a characterization of a literary text. In different genres one might suspect a different amount of social, love and family relations.

Manual annotation of relations in literary fiction is a cumbersome task. The main reason for this is that explicit mentions of relations are sparse throughout the text (especially when family and love relations are to be annotated) and the annotator has to go through many pages for sometimes just one or not even a single indicator of a relation. There are exceptions, e.g. in the fragment Aston Louise - *Lydia* in DROC, there is about one annotated relation per sentence. This sparsity has shifted the entire field of relation detection, instead of manually labelling large amounts of text, either text snippets are filtered before annotation, for example by using Active Learning [Surdeanu et al., 2012], or the data set is even annotated using *Distant Supervision* ([Craven et al., 1999] or [Mintz et al., 2009]) which is at the core of libraries such as Snorkel [Ratner et al., 2017] or DeepDive [Shin et al., 2015] or newer versions of it that are summarized as *Data Programming* [Ratner et al., 2016]. On the positive side, many of these data sets are

[1]The results of this section have been published in [Krug et al., 2017a] and [Krug et al., 2017b]

available for public use[2] and especially during the SemEval challenges, the latest being of 2018 [Gábor et al., 2018], the newest techniques are contested against each other.

This chapter is structured as follows: First, the related work of techniques for relation detection are listed, followed by a brief recap of the data sets that are used for the experiments in this chapter. In section 9.3, the different experiments towards the detection of family, social and love relations are shown, and in section 9.4, the experiments towards interaction detection are presented.

9.1. Related Work for Relation Detection

The works of [Jung et al., 2012] and [Bach and Badaskar, 2007] present an overview of the literature on relation extraction. Successful methods have been developed for the supervised and the semi-supervised case. Relation extraction usually resembles algorithms that get as input two (or more) references to entities (these pairs of references are then called instances) and their task is to predict a class with an according label which represents the relation of these references. Most experiments have been done on English text and the data set from the Automatic Content Extraction (ACE) Workshops 2004 [Doddington et al., 2004]. On the 2004 dataset, experiments that distinguish between 5 and 27 different classes like Employment, Physical, Social, Affiliation or Discourse relations (with some being subclasses of each other) have been performed. There is a huge number of approaches for this task, but they all try to get a discriminative description of the instances and then follow to classify based on these descriptions:

- Feature based classification does this by enhancing the instance (which is usually just two references to entities) by a feature vector with sometimes more than a million dimensions and classifies with methods like Maximum Entropy Models [Kambhatla, 2004] or Support Vector Machines [Jiang and Zhai, 2007]. On the ACE-2004 data the latter approach reported an F1-score of 72.9% for the detection of 7 relations. In the experiments in section 9.3, the generic features for the feature-based approaches are similar to the ones used in [Kambhatla, 2004].

- Kernel-based classification is widely used for relation extraction and has proven to yield competitive results ([Zhou et al., 2007], [Zhang et al., 2006] or the work of [Zhao and Grishman, 2005]). While feature-based methods directly enhance the representation of an instance, kernel-based methods act a little different. From an engineering point of view, a kernel can be seen as a function that gets as input two raw instances (which is a pair of references) and directly calculates a score, based on the "similarity" of these instances, a higher score resembling a higher similarity. A multitude of kernels for relation extraction have been proposed; an in-depth analysis and explanation of those is given in [Jung et al., 2012].

- Rule-based classification uses a human comprehensible representation with either

[2]`https://github.com/davidsbatista/Annotated-Semantic-Relationships-Datasets`, accessed 20.05.2020

hand-crafted or learned rules. Advantages are the inherent explanation capability and the easy integration into feature-based machine learning algorithms.

- Deep Learning Approaches: Deep Learning for the extraction of relations is based around the intuition, that in order to extract useful features, the text surrounding previously detected entities has to be embedded using a suitable network structure. For the extraction of N-gram alike features, a hierarchical Convolutional Neural Network can be used [Zeng et al., 2014]. Alternatively or additionally, a representation using Bidirectional LSTMs can be applied for the extraction [Zhang and Wang, 2015]. Since even LSTMs tend to forget information over longer sequences, an Attention Mechanism is added in modern approaches [Zhou et al., 2016].

9.2. Recap of available Data Sets for Relation Detection on German Novels

This section briefly recaps the data sets that are present in section 4.3.1. This chapter deals with the extraction of relations in German novels. For this purpose, during the Kallimachos project, we labelled 3 data sets with the label set described in chapter 4. That is a relation has one of four broader categories (Family Relation, Love Relation, Social Relation and Professional Relation). Two data sets have been labelled using Active Learning and the third one consists of 213 summaries of novels from the Kindler Online Lexicon. These data sets allow for a comparison of different methods for an automatic relation detection component, the experiments can be found in section 9.3.

Additionally, for the detection of interaction between references, 57 documents of DROC have been labelled with relations of that sort. This involves two broad categories, namely *Interaction* and *Observation*. The first category is used, whenever two entities act with each other, so that both are aware of that interaction, while the second category is only used if only one involved entity is aware of the interaction.

9.3. Experiments for the Detection of Family, Social and Love Relations

The detection of family relations as well as love relations allows for a labelled edge in a character network. While family relations remain static throughout the text, love relations might change and it is therefore required to not only extract a single label for a relation between two entities but to track them through the entire text. For this purpose, the classification scenario was selected to be as follows:

- Relations can only be binary
- Relations are only extracted between (previously detected) character references that appear in the same sentence

... wenn sie von **ihrer Mutter** gefragt wurde ... [a]

[a]... when asked by her mother ...

Figure 9.1.: An example of a family relation that is expressed with a simple chunk. Example is taken from Louise Aston - *Lydia*

- Relations are directed, that is there is a different label between two references, if the direction of the relation changes (e.g. *hasFather* might be inverted to *hasDaughter*)

For the extraction, first a rule based algorithm is presented, followed by classical feature based machine learning algorithms and a kernel for a Support Vector Machine. The section concludes with the results that are achieved using Deep Learning.

9.3.1. Rule Based Extraction

The algorithm is based on the assumption, that a relation is expressed using one of four different schemas:

Relation in a Chunk: The easiest way a relation can be described is using a single chunk. In this case, the relation usually starts at a possessive pronoun and ends at the head of the chunk. An example for this category is shown in figure 9.1.

Genitive Constructions: The second way of expressing a relation is by using a genitive construction. Genitive constructions can either be used as pre- or post modifier to a noun phrase. Depending on the exact phrase, the direction and agents of the relations have to be adapted. An example for this category is shown in figure 9.2.

Verb Constructions: While the first two constructions are all working on a phrase level, this construction is operating on the sentence level. The core idea is that there are frames of certain verbs that describe a relation between the arguments of that verb. Figure 9.3 shows two examples of such constructions. Relations that are expressed in this manner can be very hard to extract since they require not only a correct parse but might also require additional semantic information about arguments of the verb.

Speaker and Addressee: The fourth category is different compared to the previous ones, as it might operate between sentences. If an addressee inside a direct speech is found to be a relation keyword (such as ”Geliebter” (beloved) or ”Mutter” (mother)), a relation between the speaker and the addressee of the direct speech utterance can be introduced. Two examples are shown in figure 9.4

Based on these four possibilities for a construction of a relation, the presented rule based approach is of a generative nature. It starts by extracting all pairs of character

... zugleich gemeinsam mit **Quijotes Nichte** und...[a]

... als **Diener** eines so großen **Ritters** ...[b]

[a]... at the same time together with Quijotes niece and ...
[b]... as the servant of such a great knight. ...

Figure 9.2.: Two examples taken from the Kindler summary of Miguel de Cervantes - *Don Quijote*. The first sentence shows a family relation using a genitive construction as pre modifier and the second example shows a social relation that is expressed using a genitiv construction as post modifier.

... weil **ich Dich** zu sehr liebe...[a]

... **der** ebenfalls durch die Lektüre der Ritterromane verwirrt ist, über ein Problem aus dem Amadís-Roman mit **Quijote** in Streit gerät und ...[b]

[a]because I love you too much.
[b]... who is also confused by the reading of the novels of chivalry, gets into a quarrel with Quijote about a problem from the Amadís novel, and ...

Figure 9.3.: Two examples for the expression of a relation by the usage of verb constructions. The first one is taken from Louise Aston- *Lydia* and shows a love relation, and the second example is taken from and shows a rather complicated expression of a social relations. In the first example, it suffices to analyze the involved entities and the verb, while in the second example additional constraints based on a prepositional object (*in Streit*) have to be analyzed.

... »Richard« – sagte **sie** leise. – »**Geliebter**! warum fliehst Du mich?«...[a]

... als **sie** plötzlich, nach dem Fenster zeigend, ausrief: »Mein Gott, was ist das? Sollte die **Mutter** kränker geworden sein? ...[b]

[a]»Richard« - she said quietly. - »Beloved! why are you fleeing me?«
[b]when suddenly, pointing to the window, she cried out: »My God, what is that? Should the mother have become sicker?

Figure 9.4.: Two examples for the expression of relations that are expressed using speaker and addressees. Both are taken from Louise Aston- *Lydia*. The first example shows a love relation and the second one depicts a family relation.

references that appear within the same sentence. For each pair of references, a number of *representations* of that pair is determined. A rule is formulated as a representation for such a pair. So whenever a generated representation of a pair is found to be equal to a manually formulated rule, then the action of that rule determines the relation that is created. The representations that are generated for each pair of references are built based on the tokens between them, as well as the shortest dependency path (which is also an ordered list of tokens) that connects the references. The representations that are generated are built by search over the representation of the involved tokens for each pair. This is best demonstrated using an example. In figure 9.5 a chunk is depicted. For each token in that chunk, at least two alternative representations are given. The first representation is the text of the token itself, the second representation is either a blank (which is depicted using underscores) or for the character references itself the tags *NE, NE_NN, Mutter, Family_Relation.* Once all representations are added to the tokens, the rule *Mutter* (which is just a single element) is compared against these representations. Once there exists a path through all tokens, that matches the rule, then its action can be applied. For this path, all blanks can be ignored, so that the path which matches the rule is *blank, blank, blank, blank, blank, blank, Mutter.* Additional paths are added based on the dependency tree of the sentence, which is especially useful, when relations that require verbs are to be considered. In total, about 450 of such rule patterns are utilized for the extraction of these relations. They are listed in appendix D.1.

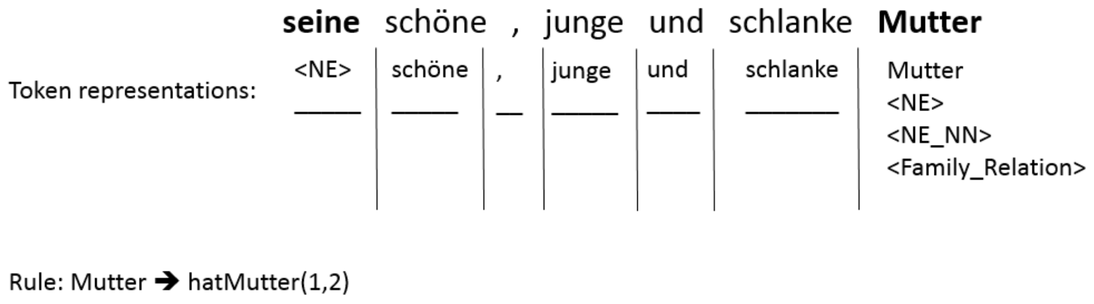


Figure 9.5.: An example noun phrase and the representation for the involved references and tokens in between the references. The action *hatMutter(1,2) is later translated, so that the first character reference is set to be the first agent and the second argument is set to be the second agent.*

From this rule set, corresponding to the first two of previously presented categories a set of three core rules stands out. The first core rule extracts relations in chunks which contain possessive constructions and two further core rules deal with the two cases of extraction in genitive constructions.

9.3.2. Extraction based On Classical Machine Learning

As already seen for the CRD, coreference resolution as well as the speaker detection, during this work, the MaxEnt classifier as well as the SVM usually yield good qualities when applied to text mining problems. For this task, similarly to the rule based approach the input to the classifier is a pair of references, for which a number of features are extracted which are finally used for the classification. The label for the classification is either ”Relation” or ”No_Relation” in the binary case, ”Love_Relation”, ”Family_Relation”, ”Professional_Relation” or ”Social_Relation” in the broad category case or one of the 57 fine grained labels in the most granular case. The feature set is inspired by the literature and contains the following features:

Reference Text: The texts and lemmata of both references, as well as the concatenation of both texts (5 binary features).

Reference POS Tags: The POS tags of each individual reference, as well as their concatenation (3 binary features).

Chunks: The text of the covering chunk of the second reference, as well as the information, whether both references reside in the same chunk (2 binary features).

Edge Direction: The direction of the edge, if the first reference resides before the second reference, then the edge is considered to be directed to the right, else to the left (1 binary feature).

Dependency Heads: The text and POS tags of the dependency heads of both references (4 binary features).

Distances: The amount of tokens and and references in between the references. If the edge is directed to the left, these number are negative.

Dependency Paths: The shortest path in the dependency tree is extracted and each token of that path is converted into its text, its POS tag and for each edge, the dependency relation is taken. The strings of these features are concatenated, so that three features (one for the joined text, one for the POS tags, and one for the dependency relations) are formed.

Genitiv The information, whether the first or the second reference is in genitiv (2 binary features).

Construction One feature for each of the references, uses the algorithm for the detection of the construction (see section 6.3.2 for the explanation of that algorithm) of the reference and uses that representation as a feature.

Additionally, inspired by the success of the rule based algorithm, the rules have been converted into additional features as follows:

Rule Matches: All rules (of the previously presented rule set) that match for a pair of references are converted into a binary feature. Additionally all actions of the matching rules are converted into a binary feature as well.

Core Rules: For each of the core rules, additional features are added to the classifier (this covers relations in the chunk, relations that are expressed using genitives and relations that are expressed using verb frames.)

The results of these experiments are depicted in table 9.1.

9.3.3. Extraction using a custom designed Kernel for the Support vector Machine

The idea of the development of a custom kernel originates from the prior success of kernel classifier for the extraction of relations, see for example [Jung et al., 2012]. Inspired by the creation of the rule based algorithm, the kernel that is presented in this section - referred to as representation kernel - is centered around the amount of common paths in the representations between two reference pairs. This section does first start with the explanation of the kernel and continues with the justification, why the kernel can be considered a valid Mercer kernel. The results are presented along the other approaches in section 9.3.5

Similar to the rule based approach, a pair of references gets enriched (on a per token basis). But since a machine learning algorithm is capable of using a large amount of features as long as the computation feasibility is guaranteed, more different aspects of the tokens are added to their representations. They include:

- Text: the pure text of a token
- Lemma: the lemma of the token
- POS: the POS Tag of the token
- Chunk: information about the covering chunk (that is B-NP, I-NP, B-PP, I-PP or O)
- GermaNet: the semantic category of the token according to GermaNet
- DepRel: the dependency relation of this token to its head (or root)
- Word2Vec: the clusterId of a pretrained Word2vec embedding file.
- NEType: only for character references: ”name” if the token is a name, ”np” if it is part of a nominal phrase or ”pron” if it is a pronoun.

So in total there are eight representations per token. The kernel itself is a function which takes two of these enriched pairs as its input and outputs a *similarity score* between these pairs, the higher the score, the more likely is it that the pairs share the same label. Of course the details have to be determined by the algorithm later on.

The similarity score between two pairs is calculated as follows: It is the amount of common *paths* between the pairs. A path means that there is at least one common representation on each token in both reference pairs. Even though the amount of possible paths p grows exponentially, it can be calculated very efficiently since it can be calculated from the amount of common representations at the token level, that is:

$$p = \prod_i \text{amountCommonRep}(\text{pair}_1(\text{i}),\text{pair}_2(\text{i}))$$

An obvious flaw of this formula is that it can only calculate a similarity score if both representations are of equal length in terms of their tokens. This is why every pair of references gets not just one, but a multitude of different representations assigned. On top, during the matching, some token representations allow a skip, so that the token at position i might be matched to a later position in the other pair. The set of different representations for each pair is derived as follows:

1. Starting with the initial representation as explained above
2. Drop all tokens in the representation that act as leafs in the partial dependency tree between the involved references.
3. Store a representation for the remaining tokens
4. Repeat step 2, as long as no involved reference acts as a leaf in the partial dependency tree.

Having a set of representations for each pair of references, the similarity score between two pairs can now be set to the maximum amount of paths that are common between any pairing of the representations (or 0).

In order to be a valid kernel, the above mentioned procedure has to satisfy the Mercer's condition [Minh et al., 2006]. That is, for every possible vector g it holds, that:

$$g^T \cdot K \cdot g = \sum_i \sum_j c_i \cdot c_j \cdot k(x_i, x_j) \geq 0$$

$k(x_i, x_j)$ being the presented kernel, K is the so called Gram matrix which stores all kernel evaluations between all available instances, x_i is a pair of references and c_i and c_j being random coefficients.

We use the fact, that the dot product (also called linear kernel) is proven to be a kernel. We then define a mapping $\Phi(x_i)$, so that, every possible path of the available vectors x_i is mapped to an integer. In lay mans terms, after this transformation, the representation of the pair of references has all possible paths assigned in a very long vector. If the path for a given integer is available, the vector contains a 1 else it contains a 0. After this representation, the scalar product between two pairs of references yields the amount of common paths, and therefore the presented kernel is a mercer kernel.

This usage of so many different possible representations as features in classical machine learning approaches (e.g. a snippet of length 20 tokens could produce 8^{20} different representations) is either unfeasible and most classifier, even when regularization is activated, perform poorly with that many distinct features.

The kernel is implemented in JKernelMachines [Picard et al., 2013] and was executed on a server. The implementation first calculates the Gram-matrix (which contains all similarities between all instances in the training data), which results in a quadratic amount of memory space and hence limits this framework to rather small problems (about 20.000 examples), even with 256GB of memory.

9.3.4. Extraction Based On Deep Learning

[3]

Based upon the previously described structure of the instances of pairs of references, the network that was utilized creates five different span embeddings based on different portions of the sentence. The first embedding extracts features from the beginning of the sentence up to (exclusive) the first reference. The second span embedding extracts features from the first reference (the fourth uses the second reference). The third embedding extract features from the text that resides in between the references and the last parts uses the tokens after the second reference until the end of the sentence. Each of the span embeddings (referred to as sentence part representation) were formed using a BiLSTM over tokens that were themselves embedded using different pretrained word embeddings. The first pretrained embedding was derived using Word2Vec, it should capture the information which is local to the token. The second embedding originates from GloVe. As GloVe is built to capture co-occurrence rations of the entire corpus, the two embeddings should complement each other. They are concatenated using a third embedding, which is built using a BiLSTM that runs over the characters. The procedure is depicted in figure 9.6. The five span embeddings are then fed into a final hidden layer and a subsequent softmax predicts the label for that instance (see figure 9.7).

Instead of just using BiLSTMs, the extraction of features using Convolutions Layers was examined. On top, the influence of two BiLSTM layers with and without an Attention mechanism was tested. All results can be found in section 9.3.5

9.3.5. Result of the Relation Detection using the different approaches

This section provides the results of the aforementioned approaches for the detection of the four labels for relation detection (family relations, love relations, professional relations and social relations). The classifier were all used in a 5-Fold scenario and the result is reported using the averaged scores over these folds.

For the detection of the family relation, the Precision of the core rules results in over 96% with a recall of more than 50%. This means, that these rules can be applied and

[3]The results of this section were obtained during a masters internship of Nils Wehner [Wehner, 2018].

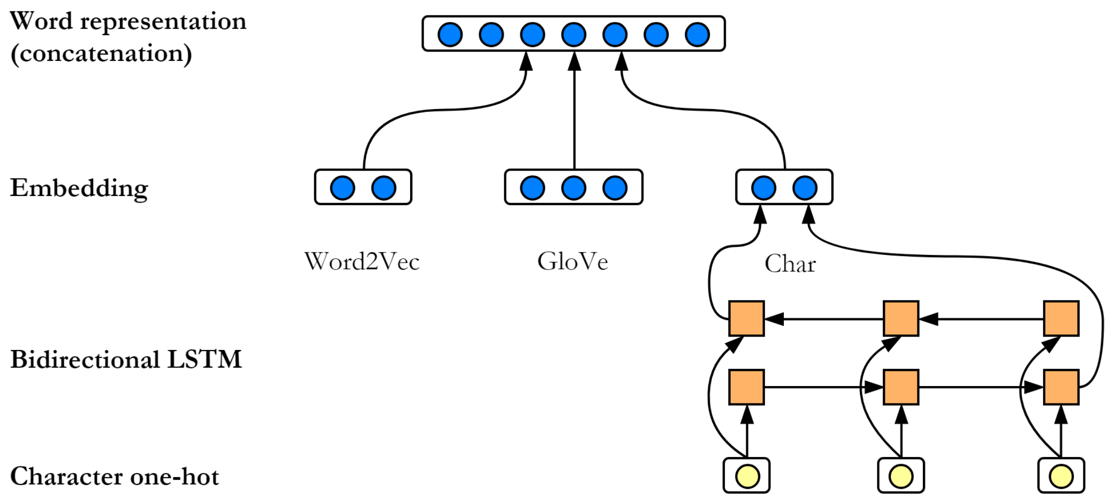


Figure 9.6.: A token is represented using a concatenation of pretrained GloVe, and Word2Vec embeddings. As third component character embeddings based on a BiLSTM layer is added. Image take from the work of Wehner.

their results seem to be reliable. Since it is expected that family relations remain constant and are repeated throughout the text, this is the suggested way of detecting them, even though the combination with the MaxEnt and the SVM with the representation kernel outperform the core rules. The detection of the love relations was successful with about 56% F1, using the combination of the MaxEnt and the rule features, but no system can produce results with a high precision. The detection of the professional relations was worst, with the highest system only obtaining an F1 score of 32%. This is probably due to the imbalance in the data, with only about 10% of every positive instance being a professional relation. The most complex neural network clearly outperformed the other approaches when it comes to the prediciton of social relations, but the total success ratio is still rather low with an F1 score of just 44.1%.

9.3.6. The Dilemma of data sets for Relation Detection

For the coarse relation detection which involves just four categories the available data set can be grouped into three distinct sets.

Novel data set: About 1100 snippets, each only spanning one sentence, labelled by active learning

Summaries I: About 1300 snippets each with a length of one sentence. Labelled using an Active Learning strategy. This data set and the Novel data set were labelled by the same annotator

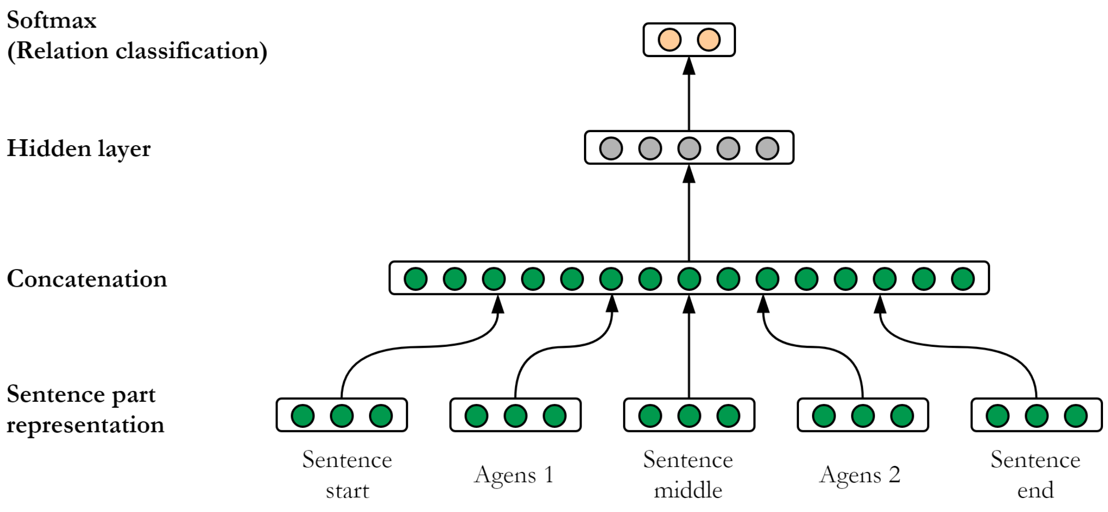


Figure 9.7.: The top layer of the network architecture for the detection of relations. The five span embeddings are concatenated and fed through a final hidden layer. Image taken from the work of Wehner.

Summaries II: This data set is labelled by a second annotator and spans 213 full summaries.

The inter annotator agreement can be determined using Summaries I and Summaries II, since all sentences of Summaries I are contained within Summaries II. More detail is given in section 4. To recap the results, an Inter annotator agreement of only about 55% F1-score could be measured between our annotators[4]. At the current state, it is assumed that this is caused by the way the data sets were labelled. Only seeing a single sentence without context causes the annotator to create less annotations than when the entire context of a text is available (only about 0.52 annotations per sentence compared to 0.75 when the entire context is available). Training the MaxEnt classifier that is enriched by the rule features on one data set and evaluating on another shows, that the agreement between the human annotator and the algorithm (slightly) exceeds the inter annotator agreement between our two annotators (having an F1 score of about 60%).

Nevertheless, by having the full novels and the corresponding summaries available a further question can be answered: Is it possible to train a relation classifier on just summaries and evaluate it on novels while still retaining about the same quality in the evaluation. Therefore, after training on each of the summaries individually and evaluating on the Novel data set, the hypothesis seems to manifest, that in order to obtain a relation detection it seems to be possible to either label summaries or novels, whatever is better accessible. Labelling summaries also has the advantage, that no section with a high density of relations has to be found in the novels (as in our case using Active Learning). More detailed results are given in table 9.1

[4] A true positive in this setting is measures whenever two annotator annotated the same span of the relation as well as the same coarse label and arc-direction

Table 9.1.: The different approaches for the detection of the family and love relations. All results are given in % and the highest scores are highlighted in bold. This evaluation assumes, that the candidate character references are available.

Approach	Family Rel			Love Rel			Professional Rel			Social Rel		
	P	R	F1	P	R	F1	P	R	F1	P	R	F1
Core Rules	**96.2**	54.8	69.9	-	-	-	60.0	16.7	26.1	-	-	-
All Rules	90.1	68.8	78.0	52.5	**55.2**	53.8	66.7	11.1	19.0	57.1	17.8	27.1
MaxEnt	76.6	63.4	69.4	**71.4**	26.3	38.5	50.0	10.5	17.4	38.9	15.6	22.2
SVM	61.1	59.1	60.1	50.0	13.2	20.8	50.0	5.3	10.0	38.9	15.6	22.2
MaxEnt$_{Rules}$	91.4	68.8	78.5	60.6	52.6	**56.3**	**66.7**	21.1	**32.0**	**69.2**	24.4	31.0
Kernel	88.2	**72.0**	**79.3**	58.3	36.8	45.2	**66.7**	10.5	18.2	55.0	24.4	33.8
BiLSTM	74.8	56.3	63.9	46.1	20.7	27.6	47.9	20.0	26.3	46.1	30.4	36.2
2 xBiLSTM	76.4	58.4	65.8	60.3	22.2	31.0	40.8	15.8	22.4	52.4	33.6	37.7
(2 xBiLSTM)$_{Att}$	72.6	60.4	65.6	51.9	17.8	25.5	56.3	**22.1**	26.8	53.1	**38.8**	**44.1**

9.3.7. Conclusion of the Preliminary Experiments

The experiments show, that the extraction of family relations apparently is the easiest among the four major labels, followed by the extraction of the love relations. Family relations usually remain stable throughout the text and are repeated multiple times, while love relations might be unstable, so the extraction by the usage of a small amount of rules with very high precision can be considered a good idea, when only family relations are to be extracted. The extraction of the love relations on the other hand is problematic, since even the combination of the rule based approach and the MaxEnt classifier only yields a quality of about 50%, for both, the Precision and the Recall. The other categories are not only problematic, when it comes to the labeling, also the extraction quality is lacking far behind. This might occur due to the much smaller amount of labelled examples for these categories. The combination of the rule based and the MaxEnt classifier showed the overall best result, clearly outperforming the approaches based on Deep Learning for family and love relations.

One rule (the one which makes use of speaker and addressees) can not be evaluated using this approach, since the data set only comprises snippets that span a single sentence. One further issue that is ignored in this setting, is that relations can be suggested based on the name of characters themselves. If two character share the same last name, either a marriage relation or a family relation is very likely. An evaluation that could take this information into account would require a data set that acts on entity level, and at the current point in time, no such data set exists.

The good results on the family relation category still have to be taken into account with a grain of salt. Most of them are expressed using chunk or noun phrases, but the information of that relation can only be used to its fullest extent if the involved entities are correctly identified by the coreference module. This however is a very hard task, because it usually involves a pronoun and a noun phrase, and both have to be resolved

Table 9.2.: Comparison of hierarchical classifiers with different structures and the standard classifier. The standard setting (flat classifier) performed worst, and all hierarchical (no matter if arranged in two or three layers) outperformed the flat classifier. This data is taken from the thesis of Fischer [Fischer, 2017]

Classification setting	**Interaction**	**Observation**
2-Layer	0.4618	**0.3250**
3-Layer	**0.4653**	0.3249
Standard	0.4516	0.2947

correctly. In section 10 it becomes apparent, that the current algorithm has issues with especially these kind of resolutions. So it can be summed up, that the relations, which can be detected easiest, are also the least useful, because to fully use them a good coreference needs to be available.

9.4. Experiments for the Detection of Interactions for Social Network Analysis

[5]The automatic extraction of interactions between character references allows for a detailed display of relations between entities in a social network. This section recaps the most fruitful approaches, their results and implications for the social networks.

When it comes to the extraction of interactions and observations, the best classifier (a linear chain conditional random field) using a similar feature set to the one presented in section 9.3.2 managed to achieve an F1 score of about 51% for the interactions and about 41% for the observations in the data set in a 5-fold cross validation scenario. It has to be noted, that class imbalance in the training data was leveraged using oversampling.

Another finding of her thesis was, that whenever the label set is of hierarchical nature (which is the case for most relation detection tasks), the usage of dedicated hierarchical classifier has a good chance to outperform a simple flat classification which directly predicts the label. In the case for the interaction detection, each hierarchical placement of different classifier yielded an improvement. The results are summarized in table 9.2

The error analysis showed, that the automatic system was more successful, when the dependency paths between the references is small. An overview is shown in figure 9.8

The implications for the automatic creation of a character network was analyzed as well, and it was found, that even though the automatic extraction only shows qualities of about 50%, the automatic networks were evaluated better, than the counterpart baselines that create edges based on co-occurences in a sentence or a paragraph. The exact results for 40 documents of DROC is shown in table 9.3

[5]This section presents results that were created during the masters thesis of Elisabeth Fischer [Fischer, 2017], that I supervised

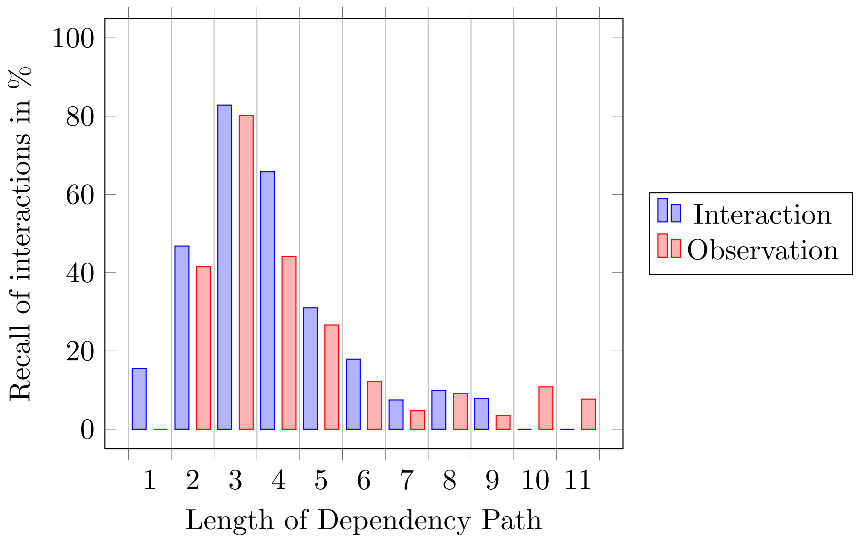


Figure 9.8.: Overview of detected interactions by path length for the different interaction types. Only takes into account whether an interaction was found at all and not whether the correct label was selected. Data was taken from the thesis of Fischer [Fischer, 2017]

Table 9.3.: Comparison of the mean Euclidean distance of system and baselines to the top nodes (according to the node degree) of the gold network, numbers taken from the thesis of Fischer [Fischer, 2017]

Name	**Highest node degree**	**Top 3 nodes**	**Top 5 nodes**
Sentence-BL	**21.1**	33.85	38.36
Paragraph-BL	32.52	50.03	60.35
Automatic Interaction detection	23.90	**31.97**	**35.21**

10. Automatic Extraction of Character Networks

This work focuses around the automatic extraction of character networks from historic novels and different ways to measure their quality, based on manually labeled data as well as expert summaries to the according novels. Character networks are a way to represent the content of the text by depicting the main characters and their relations, an aspect which might change throughout a text. An excerpt of a character network for Effi Briest is shown in figure 10.1

The extraction of character networks usually involves a preprocessing pipeline with multiple steps. The first step detects the references of the characters, the second module clusters these references into entities (coreference resolution) and additional modules are responsible for the modelling of the *relations* between characters. In this chapter, relations are either extracted as simply co-occurences of characters in the same discourse units (sentences or paragraphs) or whenever they are communicating with each other in the form of direct speech utterances. The label between entities is extracted using a separate relation classification component which acts on the sentence level.

With tooling at hand that is able to automatically extract the required data to create a character network for a given text, this allows to derive a fingerprint for a novel based on the constellation of its characters and their interactions. The true uses of such a fingerprint have yet to be studied but it would allow for experiments related to several important literary questions, such as:

- Can the network be utilized to separate novels of different genres, and therefore help in retrieval or recommendation tasks?
- Or the inverse - Can a more distinctive definition of a genre be derived, knowing both the genre and the character network?
- Does the Character Network help to examine the complexity of literary texts?

Besides great literary interest, it could enable data warehouses to answer more involved queries, e.g. "Show all texts, which are focused around a single character". Such a query can be served with the help of the networks. It means, that all texts in which the graph of the network resembles a star are of interest to the user.
Despite great effort of the community to automatically extract the networks, it still remains unclear, how the quality of such a network can be measured. The obvious way, by presenting it to a domain expert is infeasible due to time restrictions. Also, it remains unclear if a domain expert is capable of evaluating abstract measures such as the *strength of an edge* or the *size of a node*, in relation to the other edges or nodes. For

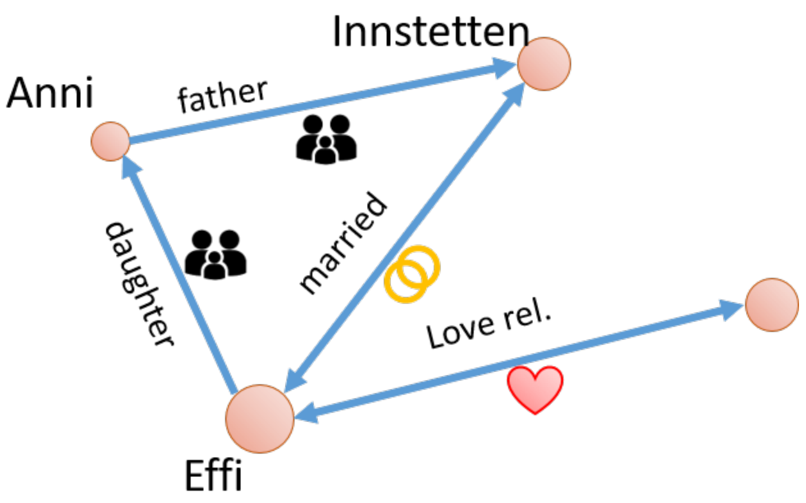


Figure 10.1.: Excerpt of a character network for the novel Effi Briest. The nodes show the main characters and the edges show coarse relations between them. This network was extracted manually.

the evaluation of the system SINNET, Agarwal et al. [Agarwal et al., 2013] made use of metrics of social network analysis (SNA) [Trilcke, 2013], such as the weighted node degree and compared gold data to system data. This enables a more streamlined process which can be evaluated automatically, but there is no clear way of interpreting these numbers, when striving for a qualitative evaluation. In this work, we used manually labelled expert summaries as proxy for real gold networks. We experimented with different ways how to use them to evaluate the nodes as well as the edges of the automatically extracted network.

We applied the text processing pipeline that was developed during the Kallimachos project [Schmidt, 2016] onto about 200 novels and show the strengths and weaknesses of the current system. Since it is especially the coreference resolution module which poses the most problems (see section 10.5.3), we adapted our coreference system in a way, that the results that are visible in the network (that is the nodes and edges between the most central characters) are treated with a higher priority and as a result are more reliable. This adaptation is done by laying more focus onto a segmentation of the novel into smaller chunks of text, applying the coreference on these local chunks and then recombining the individual results in a global coreference pass. The contributions of this chapter can be summarized as follows:

- A new method for an automatic generation and evaluation of character networks based on expert summaries of German novels is presented, and it is shown how this method can give details about the quality and major problems of the preprocessing.
- A two stage approach for the coreference resolution is presented, which uses a first

local coreference step and by recombining these local solutions in a global pass, the resulting network ends up being more precise.

- Different ways for the automatic extraction of character networks are compared, using our new metrics and previously established methods.

This work is structured as follows. In section 10.1 we give an overview over previous works that dealt with the automatic extraction of character networks. The data that is used for the experiments in this work is presented in section 10.2.Before we describe our methods for the automatic evaluation of character networks in section 10.4, section 10.3 describes the general foundations for an automatic evaluation of a character network. Section 10.5 introduces the evaluation methods based on expert summaries, and section 10.6 provides preliminary experiments towards the labelled evaluation of the edges of the character network. We conclude this work in section 10.7.

10.1. Related Work

Approaches to automatic extraction for social networks from literary text using Natural Language Processing techniques have been manifold. We categorize the different approaches with the following criteria, a summary is given in table 10.1:

- The required preprocessing
- The method to create the nodes in the network
- The methods that create the edges of the network
- The ways the researchers evaluated their networks

The system of Park et al. [Park et al., 2013] uses literary texts, detects the nodes using lists and models discourse co-occurency by the means of average distances between the appearances of two characters. They showed, that the character network extracted from fiction behaves similar to the intrinsic behaviour described by fiction writers. They do this by comparing the automatically extracted distribution of the node degree and edge weights to the power law. The work of Dekker et al. [Dekker et al., 2018] utilizes the Book NLP pipeline [Bamman et al., 2014] to preprocess literary texts and compared two sets of texts (modern literature vs classical literature) using Social Network Analysis but found that there is no significant difference between these two sets. Edwards et al. [Edwards et al., 2018] compared different aspects of SNA between manually constructed networks and automatically generated networks (on script of the TV show ”Friends”) and found, that for example the Betweenness Centrality Rank between the main characters of the series of the automatic networks is very comparable to the manually created ones. The work of Ardanuy and Sporleder [Ardanuy and Sporleder, 2014] used the StanfordNER component [Finkel et al., 2005] as well as a subsequent coreference resolution module in order to preprocess their texts. They experiment with edges based on interactions derived from conversations as well as simple co-occurences in paragraphs in order to extract the networks and subsequently clustered the networks to examine, whether the networks can be used as proxies for unsupervised genre and authorship attribution, but could only slightly exceed their baselines. The work of Elson et al. [Elson et al., 2010] can be see as one of the first papers that dealt with the automatic extraction of character networks. They extracted networks using NER and a name clustering (which can be seen as a very lightweighted coreference using only string matches) and validated their networks by the verification of different literary theories (e.g. they suspected that the more characters are involved in a story, the less dialogue is contained). Agarwal et al. [Agarwal et al., 2012] and [Jayannavar et al., 2015] extracted interactions between characters using relations of the categories *interact* and *observe* which they learn using a kernel Support Vector Machine. They evaluated their system based on SNA metrics, such as the Node-In-Degree-Centrality or Out-Centrality. The method of Jing et al. [Jing et al., 2007] provides a rich preprocessing pipeline including an event detection. They matched gold and system networks and evaluated the overlap between nodes and edges. Their results suggest, that the coreference resolution is the limiting factor of their analysis, only reaching F1-scores of 0.3. The work of Celikyilmaz [Celikyilmaz et al., 2010] stands out in the sense, that it operates entirely unsupervised using the Actor-Topic-Model. In their work they evaluated the quality of their system by comparing the predicted speaker on two novels. They therefore used speaker attribution as a proxy for the evaluation of their networks. Hassan et al. [Hassan et al., 2012] incorporated a polarity (edges include the information, whether the relation between two characters is positive or negative) on the edges to measure the relation between two individuals in online discussions. The work of Lee and Wong [Lee and Wong, 2016] builds conversational network and evaluates the network using the qualities of the individual preprocessing components. The work of Lee and Yeung [Lee and Yeung, 2012] integrated Person-location edges (such an edge is added if a person has been to that location) into the network which were subsequently used for the evaluation as well. The work of Waumans et al. [Waumans et al., 2015] shows, that by the usage of the clustering coefficient between networks, the fingerprint of the individual texts can be extracted and represented. Reger [Reger, 2017] experiments with character networks and compares the quality of different similarity measures of character networks (e.g. by the usage to Topic models, [Blei et al., 2003]) for the separation of novels of different genres. Summarizing this overview, while the steps that are involved in order to create a character network are comparable, the method for the evaluation of the networks is still in a heterogeneous state (see table 10.1.

10.2. Data

The data used in this work comprises 90 segments of DROC [Krug et al., 2018a], which come with gold character references, coreferences and attributed speakers and addressees. The second data source is derived by the Kindler online lexicon[1], where summaries of novels are publicly provided. In total we used 213 of these expert summaries and

[1]`http://kll-aktuell.cedion.de`, accessed 20.05.2020

Table 10.1.: Comparison of different approaches for an automatic extraction of character networks based on four criteria. The variety of ways to evaluate automatically extracted networks shows, that there is no established way for the evaluation of character networks, and therefore the results are barely comparable. Most groups decided to make use of preprocessing and used either entities or the group of the names as nodes for the networks. The edges are modelled using common conversations of co-occurences in a discourse unit such as a sentence or a paragraph.

Author	**Preprocessing**	**Nodes**	**Edges**	**Evaluation**
Park	List of names	Names	Co-oc	Complex systems
Dekker	Book NLP	Names	Co-oc$_{sent}$	Classic vs modern texts
Edwards	MD and CR	Names	Co-oc	SNA metrics
Ardanuy	NER and CR	Names	Co-oc$_{par}$	Genre/Author Clustering
Elson	NER and NC	Names	Conversation	SNA, literary theories
Agarwal	SINNET	Names	Interaction	Literary theories
Jing	MD, CR, ED	Entities	Conversation	Nodes and edges
Celikyilmaz	none	Exp. Actors	Conversation	Speaker Attribution
Hassan	POS, polarity	Sender	Replies	Attitude
Lee$_1$	Pipeline	Entities	Interactions	Edges
Lee$_2$	QA	Names	Conversation	Qualitative
Reger	NER, CR	Entities	Co-oc	SNA
Waumans	MD, CR	Entities	Conversation	Clustering coefficient

manually labelled them with character references, coreferences between characters as well as binary relations between the characters. For the automatic evaluation of the character networks we accessed the corresponding full novels from project TextGrid[2].

10.3. Foundations for the Evaluation of Character Networks

This section deals with the general problem of the evaluation of a character network. A character network can be represented using a set of nodes and a set of edges. It is therefore intuitive to rely on metrics that would compare a gold set of nodes against a system set of nodes (and the same for the edges). In the context of literary character networks, this is not easy to do. The first issue the arises is that this would require a ground truth in the first place. At the time of writing, no such resource exists, aside from some of very well known texts that are studied in school or in higher educational institutions. But even if some resources exist, they might vary in detail, as of which entity shows the required "importance" to be displayed in the resulting character network. As an example, compare figure 10.2, which is taken from [Brand, 2002] with figure 10.3, taken from Wikipedia[3].

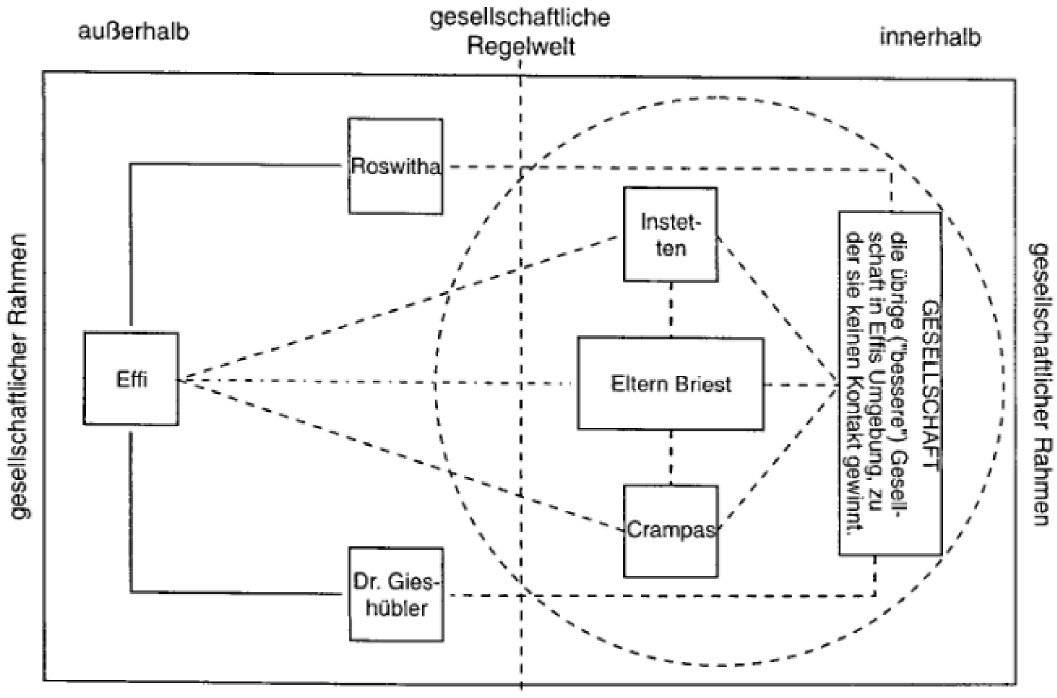

außerhalb
gesellschaftliche Regelwelt
innerhalb
gesellschaftlicher Rahmen
gesellschaftlicher Rahmen
Roswitha
Instet-ten
Effi
Eltern Briest
Crampas
Dr. Gies-hübler
GESELLSCHAFT die übrige ("bessere") Gesell-schaft in Effis Umgebung, zu der sie keinen Kontakt gewinnt.

Figure 10.2.: Exemplary character network as can be found in [Brand, 2002]

[2]https://textgrid.de/, accessed 20.05.2020
[3]https://de.wikipedia.org/wiki/Effi_Briest, accessed 18.05.2020

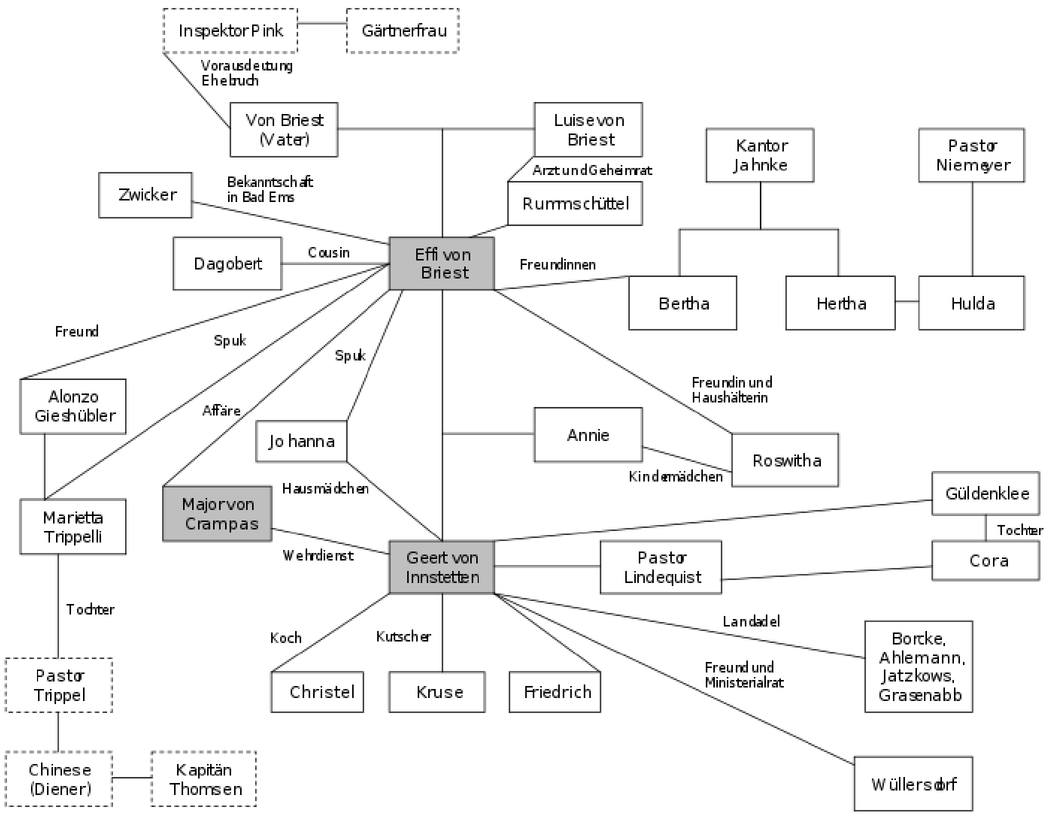


Figure 10.3.: Exemplary character network as can be found in the German Wikipedia

While the network from Wikipedia shows much more detail, the other network groups some of the minor characters into a node labeled ”society”. This leads to a conclusion that resources of a character network are highly subjective in nature and defining a single ground truth might be problematic in general. What could be done is to ask the domain expert to provide the *importance* of a character alongside the network. This would allow a ranking of the most central entities and even networks that appear to be very different could show substantial agreements. Not only could such an importance be utilized to develop a better understanding of an agreement between differing character networks, but is also allows for a more fine grained evaluation of automatic systems. An automatic system is therefore no longer required to predict exactly the same entities that are considered to be important in the ground truth. An evaluation can be done by comparing the different rankings of the entities - as provided by the ground truth - and the algorithm. This evaluation alone allows for statements such as: *”How good is the quality of the prediction of the five most central entities”*. The alignment can be measured (as it is in this thesis) for example using metrics from Information Rerieval such as Spearman’s rank correlation coefficient [Spearman, 1904] or Precision@k/Recall@k.

The previous paragraph deals with the question as of how one could cope with the different amount of nodes and edges in between solutions, but it remains to be said, when two ”nodes” should be considered *equal*. Equality occurs, when an entity of the system and an entity of the ground truth coincide. Algorithms however tend to be error-prone and the mapping of the system entities to the gold entities is not necessarily trivial. This work deals with this issue by introducing different mapping algorithms (based on some attributes of the entities, such as their names, titles etc.). A concrete example is that if a character network only has a node ”Innstetten”, but the algorithm has multiple entities with mentions that show the text ”Innstetten”. This aspect on its own could mean, that an evaluation of the character network would in fact be reduced to just another way to examine the quality of the coreference resolution. This work reports the results of different mapping algorithms (see section 10.5.2) in order to at least conserve transparency in the evaluation. This evaluation procedure can be enriched with different meta data (if available) such as the gender or other attributes of an entity.

Similar issues arise with the evaluation of edges of such a network. Instead of a manual assignment of any edge scores, a relative order (as of which edge is more important for the network) is preferred. Not only does this free the domain expert from the process of assigning any sort of absolute values as edge scores, it still retains a possibility to compare automatically derived edges based on their *relative importance*. Another bottleneck for the evaluation of the edges is that the edges depend on the choice of the nodes that are available in the network. This means, that even a good algorithm for the prediction of the edges could be rated poorly, simply because the entities are not present in the network of the ground truth. For this, the suggestion is to only consider the edges available in the ground truth and neglect any other edges predicted by the system, to not punish the system solution for a problem at a different step of the pipeline.

With these aspects made clear, the evaluation of any edge labels does not introduce any new issues. The evaluation procedure would therefore take all edges of the ground truth and the corresponding edges of the system solution and compare their label.

Summing up, the proposed evaluation procedure examines rankings instead of absolute values and a final evaluation of the network has to be a score which is formed by a node score, an edge score and a score for the labeling of the edge.

Talking about classical metrics used in Social Network Analysis. The aspect they focus is not the nodes or edges itself, but they try to capture information about the network structure itself. Not only does this ignore all problems stated above, their results are not easy to interpret. *Is a network, where the average weighted degree is 1 below the ground truth but has an average path length of 0.5 above the ground truth better or worse than a result with the opposite values?*

10.4. Methods for the Automatic Extraction of Character Networks

This section presents the different methods we applied in order to automatically extract a character network. For this work, a character network is a graph consisting of nodes, where each node represents a real world or fictional entity of the text and edges (they might be either directed or undirected) between a pair of nodes that depict a *relation* between these entities. A relation in this manner can be described in a multitude of different ways (e.g. social relation, co-presence or interactions) but for this work we decided to mainly focus around conversational relations between the entities since we have access to gold data to evaluate the quality of these relations. The first approach makes use of our full NLP-pipeline (see section 2.2), which is tailored to literary texts. A brief summary of the involved components and their required inputs and outputs:

Tokenizer and Sentence Splitter: The OpenNLP components [Baldridge, 2005] alongside their German models are used.

Morphology: We apply the RFTagger [Schmid and Laws, 2008] and the TreeTagger [Schmid, 1995] in order to get morphological information and POS-Tags.

Parser: The ParZu dependency parser [Sennrich et al., 2009] is applied and provides annotations describing the syntactical parse.

Character Reference Detection: We applied a rule based character reference detection module, described in [Jannidis et al., 2017] in order to provide character references. This component produces references in the form of names, noun phrases (which we denote as appellatives) and pronouns. For the exact algorithm, see chapter 6.

Speaker Detection: We applied the speaker detection module described in chapter 7 in order to provide information about the speaker and addressees of direct speech utterances.

Coreference: The coreference resolution module, described in [Krug et al., 2015] was further refined and tuned (see chapter 8) and applied to the according texts.

Relations: We applied our module to predict family relations, social relation and love relations, described in [Krug et al., 2017b] (see chapter 9.

10.4.1. Two-fold-algorithm

The method based on the pipeline is compared against a second algorithm, which applies the same modules but with a twist: First, the text is split into smaller segments, then the coreference resolution module is applied to these local segments (being less strict when it comes to the merges). A final global coreference resolution pass is made using the local coreference decisions and results in the final clustering which is subsequently taken for the extraction of the nodes and edges of the network. The algorithm proceeds as follows:

The algorithm starts by splitting the novel into paragraphs based on line breaks[4]. Chapters are detected from these paragraphs by the following procedure. Either by finding a paragraph that spans only one sentence and has no clear sentence end marker (such as a dot) or if the paragraph matches regular expressions containing keywords or numbers. The paragraphs are then aggregated into larger segments (which are still smaller than chapters). If a paragraph contains only quoted segments (or a frame introducing a quoted segment) or starts on a quoted segment, then it can not start a new segment and is merged with previous paragraphs(s). No segment can exceed previously detected borders of chapters. The next condition for the merge of paragraphs is based on the entities that appear in the paragraph. If no new *anchor* references are found in a paragraph (compared to the list of previous paragraphs that form a segment), then the paragraph gets merged into the segment as well. An anchor reference is a reference that is either a name or an appellative that is part of a relation, or is detected to be speaking (references inside quoted segments are not considered as anchor). The match for new anchors is done by simple string matching. At this stage of the creation of the segments, a dedicated mechanism for the detection of scene borders could be integrated, but we currently have no reliable module for them. Scene borders might be detected using triggers that reflect changes in narrative time, changes in location or changes in the character constellation (e.g. if someone leaves the scene). The goal of the segmentation into these segments is to approximate a segmentation into scenes. This stage of merging paragraphs is followed by a second stage that utilizes further constraints. If a segment which contains quoted speech only has a single anchor reference, then the segments are merged (we assume that one entity is not speaking to itself). Also, if two adjacent segments feature exactly the same anchor references, then we merge the segments as well (this might combine different literary scenes, but we still assume, that the involved entities remain the same). Having these segments, we apply a local coreference resolution to all of these segments which is very generous, when it comes to string matches, even combining strings we usually would not do (e.g. we assume that there is only one "mother" in such a segment, which is highly problematic on the document level). After the local coreference, we extract links from the resulting clusters, which we use as prior knowledge for the global coreference stage, which in turn produces the final coreference clustering. This clustering is then used for the extraction of statistics for the character network.

The main idea behind the two-fold algorithm is that a character network is a compressed form of the entities and relations of a novel, which means not all information in the original text is displayed by these networks. An example is that only the most central entities (the anchors) appear as nodes to keep the network clear. This means that the algorithm can focus on the main figures and be less precise with the other entities and their relations.

[4]In our data, this provides reliable information about the intended paragraph breaks by the author

10.5. Quantitative evaluation of character networks based on expert summaries

Evaluating a character network is a difficult task, because no easy accessible source of networks that were created by domain experts exists. This is why good proxies to this missing gold standard are required. In this work, we decided to make use of expert summaries, derived from the Kindler online lexicon (for more information, see 10.2). In total, we have access to 213 expert summaries to which the according full text novel was available. We preprocessed the full-text novels with the pipeline described in section 10.4 and applied both algorithms for the automatic extraction of character networks on them as well. This gives us access to all information that is required in order to create a character network for both methods presented in section 10.4.

The section is structured as follows. First, we give results of simple experiments towards the correlation of the amount of entities and the length of a novel. Then we present our methods for the automatic evaluation of the nodes and the automatic evaluation of the edges respectively.

10.5.1. Prior experiments towards an automatic derivation of the amount of nodes for a character network

When creating character networks, one important aspect which an algorithm needs to decide is the *amount* of nodes in the resulting graph. While in semi-automatic applications, a user can interactively choose for the *top k* entities, a nice feature would be if the algorithm could derive a suitable amount of important entities in the text on its own. For this purpose, we conducted experiments based on the full-texts and their summaries. For these experiments, we assume that the expert summaries at least mention every important character of the according novel. We then proceeded to plot the amount of entities in summaries (which are manually annotated in the documents) against the length (in tokens) of the full-text novel (see figure 10.4). This experiment shows that no easy method based on the length of the text can be found that provides a reasonable amount of nodes for the network.

What can be noted is that about 87% of all novels (185 out of 213) contain at most 20 entities in the summary, providing a reasonable upper bound to the amount of nodes in the network.

The next experiment we conducted was to select only the entities that are named in the summaries, that is we remove all entities where no single character reference represents a named entity. The results of this procedure are shown in figure 10.5.

The total amount of named entities is much lower when aggregated over all 213 documents, more precisely, the average number of entities per document drops from about 13.3 to just 4.8; in absolute numbers, about 1800 entities of 2800 total entities were removed by this filtering procedure. Even though this does not really help to determine an exact amount of nodes for the network, it strongly suggests, that aside from named

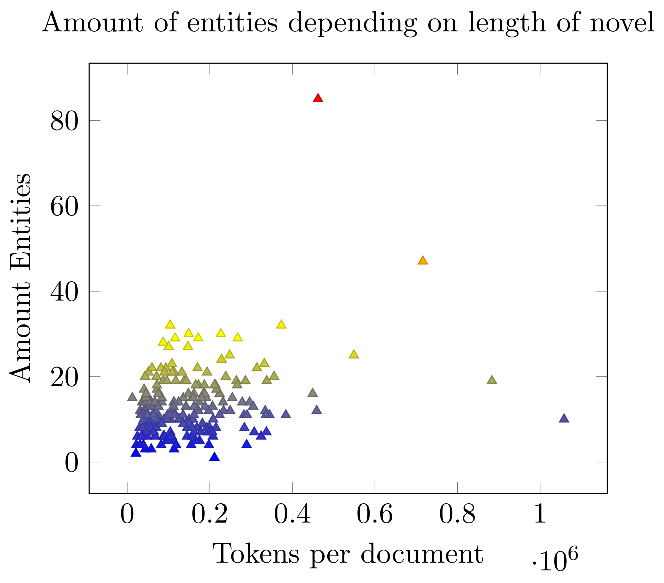


Figure 10.4.: The amount of entities (as determined by the expert summary) plotted against the amount of tokens of the entire novels.

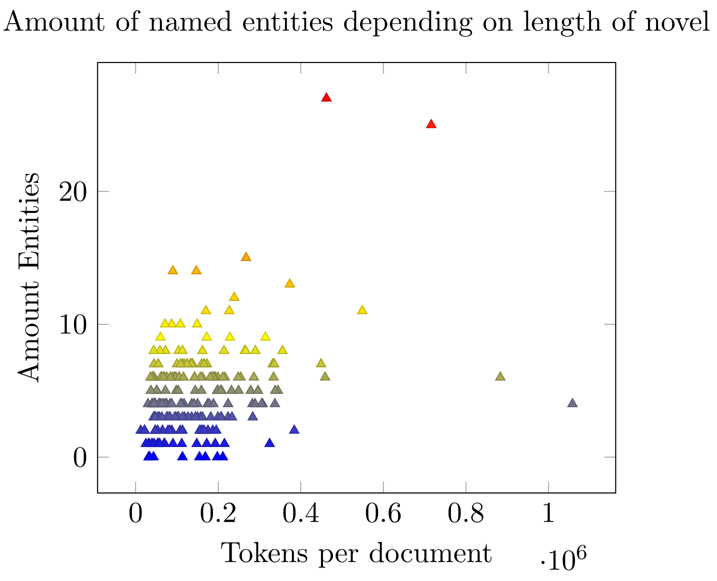


Figure 10.5.: The amount of named entities (as determined by the expert summary) plotted against the amount of tokens of the entire novels. Based on this metric, the spread in the data is even larger compared to the raw amount of entities.

characters, there are also unnamed entities that should appear in the networks. This becomes especially apparent, when considering the novel of Lewis Caroll -*Alice im Wunderland*, where *Alice* is the only named character in the summary. On the other hand, when considering the summary of *Unterm Birnbaum* by Theodor Fontane, at least two characters that are named in the novel (namely the wife of the protagonist, as well as their neighbour) are not named in the summary. For entire novels, we conclude that just based on the length of the novels, we cannot determine a reasonable amount of entities for the network. Instead, more complex attributes of a novel need to be examined.

10.5.2. Evaluation of the Nodes of the Network based on Expert Summaries

Since a reasonable amount of nodes for the network cannot be determined by simple metrics based on the length of the document alone, we used the capabilities of the proposed algorithms themselves in order to predict a reasonable amount of entities. For the evaluations in this section, we assume that all major characters are at least mentioned in the expert summaries. We therefore extracted all entities from the summaries as well as all entities from the automatically preprocessed full novels (this section explains our evaluation measures based on the full pipeline as it provides data for all our metrics) and compared the resulting sets. The first part of the evaluation did not integrate a cutoff for entities into the algorithms, while the second scenario for the evaluation of the entities involves a ranking that can be derived automatically. For the comparison we conducted different scenarios:

Bag of Entities (BoE): In this scenario, both sets of entities are not ranked but rather treated as an unordered set. The gold set (as derived from the summaries) and the system set (automatically derived from the full novels) are mapped onto each other using either a *soft* mapping or a *hard* mapping. Soft mapping means, that entities from the system set can be mapped to multiple different entities of the gold set. A hard mapping restricts this, so that after the mapping, only 1:1 pairs between a gold entity and a system entity are allowed. The soft setting directs the focus of the evaluation to whether references are found at all, while the hard setting evaluates the purity of the entities (therefore the influence of the coreference resolution is higher in the more strict setting). When the pairs are created, we report the amount of entities from the summaries we were able to match and denote them as **true positive star (tp*)**[5]. A **false positive star (fp*)** occurs, when a system entity could not be matched against a gold entity and a **false negative star (fn*)** is a gold entity that could not be matched against a system entity.

Ranked Entities: In this setting, we derive an ordered list of entities from the set of system entities. These ordered lists are compared using a common performance metric for rankings. We use Precision@k to denote the ratio of correct entities in the system ranking (when using the top k system entities) compared to all gold entities, again

[5]The star denotes that these statistics are not verified against true gold data, but the performance metrics Recall, Precision and F1-score are suitable metrics for this evaluation nevertheless

differentiating between a soft and a hard mapping. In order to evaluate, whether sheer counting can produce a reliable ranking of the important entities, we created the ordering in two ways:

- **Most Frequent**: The entities are sorted by the amount of appearances in the text.
- **Most communicative**: The entities are sorted based on the amount of direct speech utterances that they are speaking.

All settings require a mapping algorithm between entities of the system result and entities of the gold standard. For the following experiments, we used three different mapping algorithms. The first mapping algorithm is denoted as *soft* mapping algorithm. For this, we use a pairwise compatibility score between gold and system entities. Each common name yields a score of +2, each common appellative a score of +1. In order to resolve equal scorings, we integrated an additional score based on the relative size of the entity compared to the total amount of references in the full novel (so if there is more than one compatible gold cluster, we enforce the pairing with the larger system cluster). The last score is only provided if the score before this point is greater than zero. The second mapping algorithm is denoted as *hard* mapping algorithm and only allows 1:1 matchings between the system entities and the gold entities. We calculated a pairwise compatibility score between the entities (of gold and system) in the same manner as for the soft mapping (this creates a 2D-array of pairwise compatibility scores) and created a hard mapping using these scores with the Hungarian method [Kuhn, 1955]. A third way for a mapping, which makes use of more semantically detailed information about entities is applied. It basically considers first and last names as well as titles of the entities. We denote this mapping as *semantic mapping.* The algorithm for this mapping considers two entities as compatible, if they agree on their first and/or their last names (as long as names are available). And if no names are available, it checks for common noble titles, spiritual titles and professions; if none of this information is available either, then it compares the remaining appellatives of the entities. It is noteworthy, that whenever either the system or the gold entity show names (or titles) but the other one does not, then they cannot be considered as compatible by later means.

The first issue we had to face when mapping the strings of the names of the summary to the full texts, was that we found that only 80% of the names could be found in the text of the novels. This issue arises mainly due to different variations of the names that might be introduced during the translation (e.g. *Antonios* instead of *Antonius* or *Varvara* instead of *Warwara*). Another source of differences is background knowledge of the author of the summary, who named characters that have not been named in the novel (e.g. Hoffmann E.T.A. - *Die Elixiere des Teufels* where *Medardus* is mentioned in the summary but never appears in the novel). We therefore filtered the summaries, so that at least the strings of the names could be found in the according full text. This left only 66 pairs of summaries and according novels for the evaluation. The mapping algorithm needs to be adjusted for the spelling variations, if the other summaries are to be used as well (we opted not to do this, since this would introduce additional noise to the results). This section does now proceed through the evaluation settings starting with the least demanding for the automatic algorithms up to the settings that are the most demanding for the algorithms.

After the filtering process of the summaries, the first experiment measures the amount of entities that can be detected with the names provided by the summary. The previous filtering process ensured that 100% of their names are contained in the full novel. The automatic soft mapping yields about 85% of all entities that could be detected by their names. This provides an intrinsic metric for the quality of the character reference detection (CRD) towards the automatic creation of character networks. The next experiment we conducted was to map the gold and the system entities using the semantic mapping. This mapping also respects more automatically derived meta data of the entities, namely the semantic categories of their tokens (e.g. first names, last names or titles). When including these criteria, the mapping score drops to a Recall of only about 69% when evaluated on the named entities and to just about 46% when being evaluated on all entities (Yet again underlining that appellatives form a more difficult group to match. It is noteworthy however, that while the summaries were filtered so that the names are contained in the full novel, we did not force that all appellatives occur in the according novels).

Since for every entity only a single node is introduced in the network, we mapped the system entities in a 1:1 fashion onto the gold entities, using the Hungarian method. If the best mapping between a gold entity and a system entity only had a score of zero, we did not allow the matching which results in a false negative afterwards. For the named entities, this results in an Recall of 92.3%, even exceeding the primitive match. This means, that even though the name of a character was not detected, the coreference still provided a cluster with appellatives to describe that entity. When using all entities of the summaries, the Recall drops to about 50.4%. This shows, that our current coreference module has major issues to produce clean clusters of characters that are not named.

Summing up, the quality of the character reference detection was evaluated with a score of about 85% for named entities and about 80% for the recognition of all entities contained in the summaries. We expect that our networks contain nodes for about 92% of all named entities (as shown by the hard mapping), but can only detect the names for 85% of all named entities. The difference of 7% is the set of characters that could only be matched by appellatives and not by the means of names. Including the results of the semantic mapping, the coreference module still has trouble to correctly resolve entities which share the same last names, seeing that the Recall dropped to about 69%.

Character networks usually depict more prominent entities with a *bigger* node. While the exact size of a node in the final representation is a cosmetic aspect, we can qualitatively evaluate the ranking of the nodes by their *importance*. In order to determine the quality of different algorithms on the (most important) entities, we ranked the system entities (using the sorting criteria mentioned above) and introduced a cutoff for them. We plotted the averaged Precision@k, starting at $k = 1$ up to Precision@20 (which was determined to be a robust upper value for the amount of entities by our experiments in section 10.5.1). A schematic sketch of this procedure is shown in figure 10.6.

This evaluation shows how reliable the top-k ranked entities of an algorithm are when

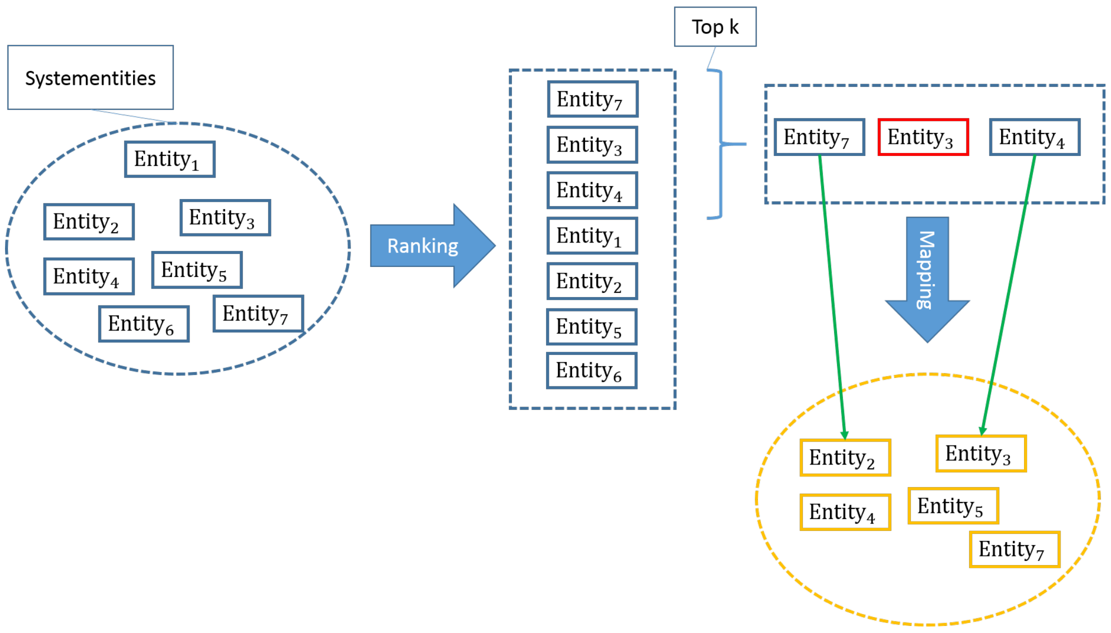


Figure 10.6.: A schematic image of the process for the ranked evaluation of the nodes in the network. All system entities are ranked (by frequence of occurence or by the amount of communication), the top-k of the ranked entities are chosen for the mapping onto the gold entities. The mapping is yet again performed either hard or soft. In this example, a successful mapping could be established for two of the top 3 entities as determined by the system.

compared to their summaries. We compared the ranking based on simple counts with a more involved method that counts the number of times an entity was assigned as a speaker of different direct speech utterances. The results for the full pipeline are shown in figures 10.7 for the soft mapping and 10.8 using the hard mapping.

In both settings, the simple counting method created a better way to find the most relevant entities, even though the difference between both methods is rather small. In all settings, the two most important entities are usually detected correctly, but the quality drops rather fast, which means that among the top five nodes 1-2 entities can not be mapped to the summary and therefore should not appear in the network.

10.5.3. Evaluation of the Edges of the Network

The second part of the evaluation of the network is based around its edges. For this part, we can make use of DROC, since in DROC we have labelled coreference information as well as the information about which entities are currently speaking. From this information, we can calculate different gold statistics:

- The amount of common occurrences (co-occurences) within a section. We experimented with sentences and paragraphs, since our data contains reliable information

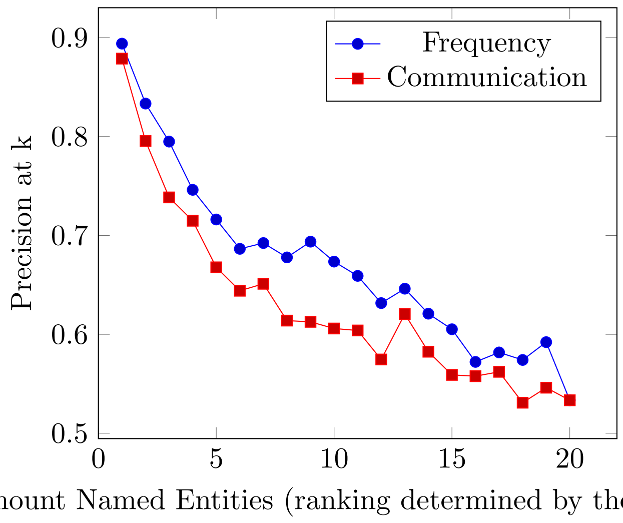


Figure 10.7.: The amount of the top k named entities of the full novel (as derived by the full pipeline) that could be mapped to the entities of the summaries using the **soft** mapping algorithm, differentiated between the ranking induced by frequency and the ranking by the number of times an entity acts as a speaker throughout the novel (communication).

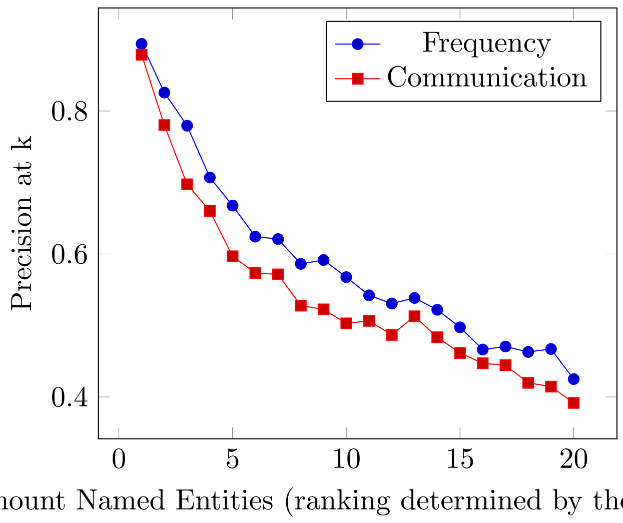


Figure 10.8.: The amount of the top k named entities of the full novel (as derived by the full pipeline) that could be mapped to the entities of the summaries using the **hard** mapping algorithm, differentiated between the ranking induced by frequency and the ranking by the number of times an entity acts as a speaker throughout the novel (communication).

about paragraphs.

- The amount of interactions based on communication between entities. Such an interaction is derived if one entity acts as a speaker and the other entity acts as an addressee of a direct speech utterance.

While neither of these metrics reflects a perfect representation of the relation between characters, at least the amount of communication between entities forms an important aspect. In most publications (see table 10.1), edges represent the *interact*-relation between characters. While there are multiple points of view for describing a relation between characters (such as their social relationship, their affinity or polarity), in this section, we restrict the analysis to just the amount of interactions as derived by the aforementioned statistics between the entities. We can then proceed to make use of some of the established performance metrics of social network analysis (following [Dekker et al., 2018]) and decided to use the following centrality measures:

Average Degree: The Average Degree is the average amount of neighbours of an entity. It represents the connectivity of an entity.

Average Weighted Degree: The Average Weighted Degree also respects the weights of the edges. A high Weighted Degree can therefore either be achieved by fewer edges with a high weight or by more edges with a lower weight.

Average Path Length: The Average Path Length is the average of all shortest paths (as determined by the Dijkstra algorithm [Dijkstra, 1959]). A low Average Path length suggests, that either many entities are not connected by any paths and/or the existing shortest paths are very short.

Graph Density: It reflects the ratio of available edges compared to the total amount of possible edges. A higher density means that all entities seem to interact with each other.

However even though these metrics have seen great use, they are not really suitable for literary interpretations but are better suited to gain more insight into the quality and especially the influence of the different preprocessing steps. This is why, during the evaluation for the full pipeline, we switched between automatically derived preprocessing and gold information for:

1. The detection of the character references (CRD)

2. The attribution of the speaker (Speaker)

3. The coreference resolution (CR)

Table 10.2.: Comparison of the SNA metrics for different amount of preprocessing and the edges weighted by the amount of co-occurences on the sentence level. We make use of the total amount of entities (TAE), average amount of entities (AAE), Average Degree (AD), Average Weighted Degree (AWD), Average Path Length (APL) and Graph Density (GD). We varied the amount of preprocessing that is done automatically and denote such steps with *sys* and steps that used the gold data with *gold*.

Experiment	Relation By	TAE	AAE	AD	AWD	APL	GD
CRD$_{gold}$**CR**$_{gold}$	Co-oc Sent	5288	58.8	4.0	8.0	0.8	0.08
CRD$_{gold}$CR$_{sys}$	Co-oc Sent	8896	98.8	4.8	6.7	0.99	0.06
CRD$_{sys}$CR$_{sys}$	Co-oc Sent	8067	89.6	5.0	7.1	1.0	0.06
CRD$_{gold}$CR$_{gold}$	Co-oc Par	5288	58.8	8.7	12.3	0.85	0.18
CRD$_{gold}$CR$_{sys}$	Co-oc Par	8182	90.9	11.8	14.4	0.94	0.15
CRD$_{sys}$CR$_{sys}$	Co-oc Par	8067	89.63	11.05	13.6	0.94	0.14
CRD$_{gold}$Speaker$_{gold}$CR$_{gold}$	Comm	5288	58.8	0.24	1.75	0.016	0.006
CRD$_{gold}$Speaker$_{gold}$CR$_{sys}$	Comm	8896	98.84	0.28	0.85	0.027	0.004
CRD$_{gold}$Speaker$_{sys}$CR$_{gold}$	Comm	5288	58.8	0.18	1.13	0.01	0.004
CRD$_{gold}$Speaker$_{sys}$CR$_{sys}$	Comm	8182	90.9	0.19	0.67	0.012	0.002
CRD$_{sys}$Speaker$_{sys}$CR$_{sys}$	Comm	8067	89.6	0.175	0.66	0.011	0.002

The results for this experiment are seen in table 10.2. We note that we cannot conduct experiments with gold algorithms applied after a system CRD, since both, the speaker detection and the CR rely on the results of the CRD.

The table shows that, whenever system coreference resolution is applied, the amount of entities is much too high (this supports our results in section 10.5.2). Both the distances of the gold solution between Average Degree and the Average Weighted Degree suggest that even though the system produces more entities, the average node degree remains stable. In all settings, the Graph Density appears to be low, with the system pipeline being able to reproduce about 75% of the edges of the gold data, when co-occurrences are measured at sentence level and about 77% when evaluated on the paragraph level. The metrics on the paragraphs are more tolerant towards mistakes in the preprocessing, since no mistakes occur, when e.g. a reference is resolved to a wrong entity that resides in the same paragraph. The evaluation based on communications allows a more detailed insight into the quality. The graph is more sparse (about one order of magnitude), which means that fewer entities are connected. When using a system coreference, the Average Degree is yet again higher than it is supposed to be, this is probably due to the coreference being unable to merge entities that belong together. This has a severe impact when considering the Averaged Weighted Degree because the scores for the correct edges are now distributed among other entities. The influence of the CRD seems to be the smallest of the three automatic systems, which means, that character references that appear close or in dialogues are found reliably. In terms of statistical significance, aside from the Average Path Length for the communication networks, all measures would be considered statistically significant different compared to the corresponding gold measures (using a paired t-test with $p<0.05$). To sum up the results, the strongest influence on the results originates from the automatic coreference, followed by the automatic attribution of speakers (at least for the communication case).

An evaluation of the character networks based on sheer counts of the edges or other SNA metrics is not promising (e.g. there is no visual evidence of detailed edge scores or node sizes in the network anyway). An approach which offers better interpretability is to compare the rankings of the most important edges with each other. For this, we used the same three ways (Co-occurences in sentence/paragraph or based on communication) to score edges between entities but this time evaluated the ranking of the edges. This neglects the absolute values of the edges but instead retains their relative importance among each other. In order to compare the gold results with those derived by the system, we applied the **hard** mapping algorithm that is introduced in 10.5.2 to map the gold and the system entities. This allows for gold edges to be found among the system edges. The quality of the rankings are calculated using the Spearman's rank correlation coefficient [Spearman, 1904]. For this, we varied the amount of most important edges as determined by the gold data (up to a maximum amount of 30 edges) and calculated the average correlation coefficient. The results for different ways to define the edges scores and different amount of preprocessing are shown in figures 10.9 and 10.10. The evaluation based on the ranks demonstrates in a much clearer way, that the more system preprocessing is applied, the worse the results become.

There is only one setting, in which we could apply the gold coreference. In this setting, the correlation between the gold and the system solution is at about 70%, when using a system speaker detection and CRD. As soon as a system coreference is employed, the results drop to a correlation coefficient of at most 50%, adding more preprocessing by automatic means also decreases the correlation. The fully automatic pipeline produces more reliable results comparing the edges based on communication that those that are created using the co-occurences methods. It is noteworthy, that in the communication setting, the correlation seems to rise after about the top 20 edges, but there is only one novel (fragment) in our setting which had more than 20 edges, so the data after 20 is not reliable. The two stage approach, which focuses on the more central entities outperforms the result that uses only system preprocessing up to 20 edges, in the beginning almost doubling the correlation, but it still can not compensate the system coreference, given that it produces correlations between 20 and 50% (as with all other results that use a system coreference) .

10.6. Evaluation of the Labelled Relations Between Entities

Instead of evaluating edges based on an edge score or different SNA metrics, a more intuitive way is by the use of edge labels which represent the relations between characters. Our pipeline contains a module to automatically predict relations between detected character references. The automatically predicted labels can have one of three labels

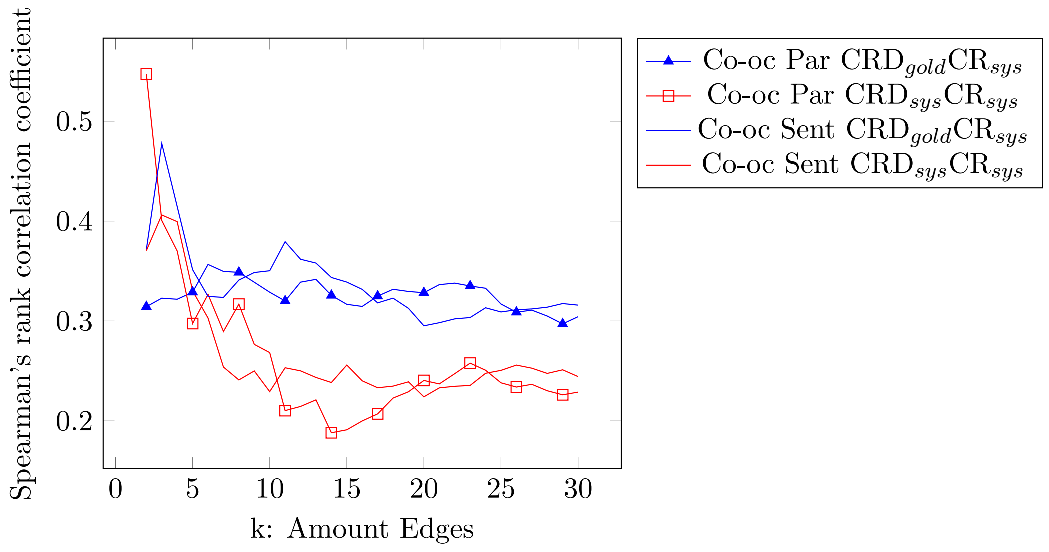


Figure 10.9.: Averaged Spearman's rank correlation coefficient using two different ways to aggregate the edge scores (co-oc Sent and co-oc Par). The subscript *sys* refers to the component being executed by an algorithm, while the subscript *gold* refers to the usage of the gold solution.

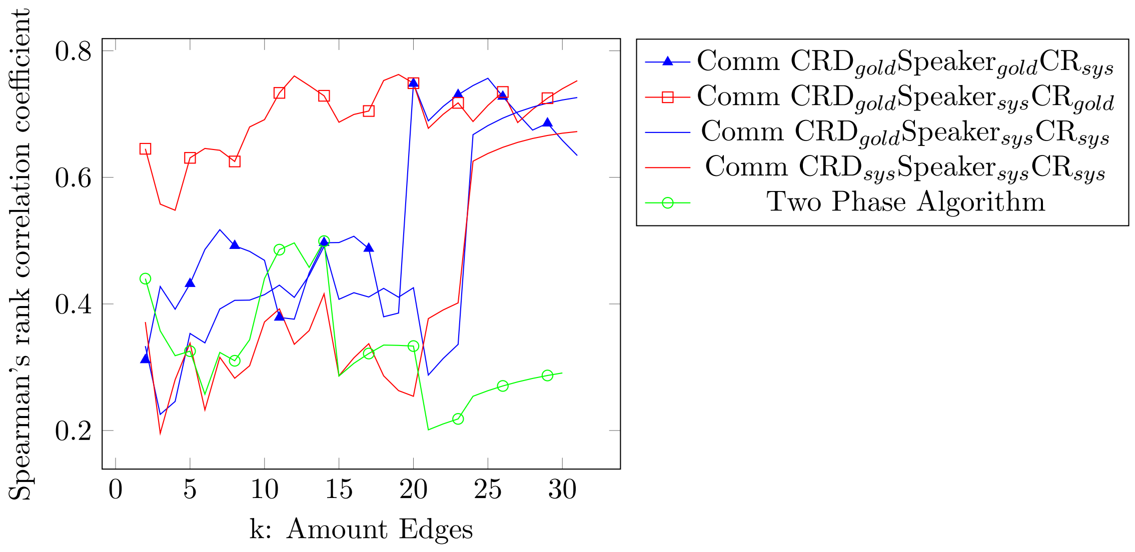


Figure 10.10.: Averaged Spearman's rank correlation coefficient for the edge scores based on the amount of communication an entity is involved. The subscript *sys* refers to the component being executed by an algorithm, while the subscript *gold* refers to the usage of the gold solution.

(FamilyRelation, LoveRelation or SocialRelation). We reused the 66 Kindler summaries where at least all name strings are contained, and used the manually tagged relations as gold data and automatically extracted relations from the according full novel. This experiment is very demanding to the automatic algorithm, it requires:

- Character references in a relation to be detected (CRD).
- Character references in a relation to be resolved to the correct entity.
- The entity of the full novel to be mapped to the correct entity of the summary.
- The correct label has to be picked for a relation between two entities.

Since most relations (about 2/3 of all instances) occur in genitive constructions ("Peter's mother") or possessive constructions ("his mother"), the detection of the character reference that indicates the relation is the easiest of these steps, because it is usually coming from a limited vocabulary (e.g. *mother*, *son*, *beloved*,...). The hard parts are the other two steps. Our experiments in section 10.5.2 showed that we are able to map about 92% of all named entities to the summary but are only able to find a mapping for about 50% of all entities, leaving, at least for the case with all entities, rather pessimistic upper bounds. For the named entity case, even though we potentially have a high mapping score, the coreference has to correctly resolve the references in the relations (which are usually appellatives). This however is one of the biggest problems for our current coreference algorithm and also one of the main issues why it is creating far too many entities when evaluated against the gold data (see table 10.2 for the effect). Then, after the mapping was successful and the references in the relation were resolved correctly to the entities, the last issue has to be resolved. Since relations are extracted throughout the entire novel, the correctly resolved relation has to be selected from all other relations that were resolved to these entities as well. For this evaluation, we neglect that the relation between two entities might change throughout the novel since our algorithms just do not show the required performance for this to really matter.

For the results in table 10.3, we used the hard mapping variant and evaluated the relation in a relaxed manner, where a true positive could be obtained when at least one assigned relation had the correct label, and in a strict manner, where the label for the relation was selected as the most frequent one that was assigned automatically. Since we can not expect that the summaries mention all relations between the entities, a false negative is awarded, whenever either no mapping could be found, no relation could be found or the wrong label is extracted, and a false positive, whenever a relation with a wrong label was found.

In all evaluation settings, the Precision is much higher than the Recall, but at the current stage - as expected - the amount of errors that propagate for this experiment can not be compensated, even though the same relation tends to be mentioned multiple times throughout a novel. The main disadvantage is that this approach relies too much on the coreference resolution.

Table 10.3.: The evaluation of automatically assigned relation labels between two entities. All values are given in per cent. In total, 138 love relations, 182 family relations and 207 social relations are marked in the gold standard.

	FamilyRelation			LoveRelation			SocialRelation		
	P	R	F1	P	R	F1	P	R	F1
Entities$_{all}$, relaxed	69.6	21.4	32.7	59.1	18.8	28.6	72.2	18.8	29.9
Entities$_{named}$, relaxed	73.3	25.6	37.9	44.4	12.2	19.0	77.4	27.0	40.0
Entities$_{all}$, strict	48.2	14.8	22.7	40.9	13.0	19.8	44.4	11.6	18.4
Entities$_{named}$, strict	35.5	12.4	18.3	22.2	6.1	9.5	35.5	12.4	18.3

10.7. Conclusion

In this work, we presented new methods for the automatic evaluation of character networks, extracted from German literary novels. Since, at least at the current point in time, an evaluation that involves a domain expert is not feasible, we presented different methods based on summaries of the novels. A summary is usually much easier to come by, compared to a domain expert. In this work we used expert summaries derived from the Kindler Online lexicon.

We presented different ways to automatically derive a quality of the nodes in the network, using either a soft mapping (which also provides a statement about the quality of the CRD) and a hard mapping. The difference in quality between the soft and hard mapping gives an insight into the quality and problems of the automatic coreference module. In our case, about 85% of all named characters and about 80% of all characters could be detected and matched to the summaries. Even though the experiments that revolve around determining a cutoff for novels of different lengths was not successful, we found (using the summaries) that about 20 characters is a reasonable amount to be represented in the network.

In order to evaluate the importance of different characters, we compared the rankings of characters of the novels and the summaries, and found that the quality of the most central entities is good, but the quality is dropping fast when considering additional characters. We found that a character network requires more than just named entities but instead also appellatives should appear in the network. We propose to use the speakers of direct speech utterances alongside the named characters for the nodes of the network.

For the automatic evaluation of the edges of the network, we used gold data for coreference and speaker attribution of DROC and used classical SNA metrics. However, they are very strict and do not really provide insight into the problems of the algorithms. This is why we proposed to make use of Spearman's ranked correlation coefficient in order to relax the evaluation. Instead of evaluating based on the hard numbers, this correlation coefficient only requires the relative importance among edges to be the same. These experiments reveal, that our coreference resolution module is still the main source of errors that appear in the network. In order to mitigate this, we proposed a two stage algorithm, which first splits the novels into smaller segments, applies the coreference on the local segments and recombines the results. Even though this algorithm outperformed the naive experiment, more refinements have to be made (especially in the recombination stage) in order to further improve the results. If summaries are labelled with relations as well, then they can also provide a source for an automatic evaluation of labelled relationships between characters. We found, that even though the extracted relations show a high Precision, due to the additional steps involved in the evaluation, the total quality did not exceed 30% for all three kinds of relations.

We especially plan to invest further energy into the development of the two stage algorithm. Partitioning the novel into smaller segments could be beneficial in a multitude of ways. The first being that the main issue of the end-to-end coreference system by [Lee et al., 2017] is that it tends to forget about entities if the segment it is applied to is too long. By splitting the documents into smaller chunks, we could apply a neural module to the chunks and recombine the results using our rule-based system in order to combine the strengths of both approaches.

11. Summary and Outlook

The chapter concludes this thesis by reciting the findings and contributions with its goal to extract character networks from German literary novels. Since the current state is still not sufficient for a fully automatic network which has reliable results, the outlook serves the purpose about the next steps that are necessary in order to improve upon the current state.

11.1. Summary

The main contribution of this thesis is the development of a pipeline which enriches plain text with necessary meta data in the form of annotations, in order to extract detailed character networks in a fully automatic manner. The development of such a component requires throughout preprocessing of the texts. This work reviews existing components and their quality on German literary texts in section 8.2.2. It shows, that both the TreeTagger and the RFTagger appear to be reliable components, even when applied on a different domain of data. This however can not be restated for the quality of syntactic parsers. The Berkeley and Stanford Constituency Parsers only reach an entity score of about 60% when evaluated on the literary domain and among three Dependency Parsers, the ParZu parser (which is the only system using a hand crafted grammar) yielded by far the best results, but can still not exceed 80% F1-score, when evaluated to predict subjects and objects. In order to extract character networks, a crucial step is to extract characters and their references in order to have a basis for the nodes in the network. Existing systems, such as StanfordNER only predict names and ditch on noun phrases and pronouns that reflect characters. An automatic adaption of this system - even though much effort was put in, could not nearly come close to results obtained by supervision. In chapter 6 it was shown, that a component for the automatic detection of character references which produces entity-F1 scores above 90% could be obtained by using classical machine learning, Deep Learning as well as pure rule-based algorithms. A combination of classical machine learning and rule-based systems obtained the maximum score of about 93%. Rule-based systems seem to have been disappeared from the current approaches of current research. For the task of Named Entity Recognition, this might be due to the way datasets have been constructed in the past. Since they are just a collection of different sentences from different newspaper articles or Wikipedia articles, one of the main strength of rule-based systems (using local and global contexts) is eliminated from the start. A presentation of the main methods based on machine learning that were used in the domain of NLP is given in section 3.2. This review includes a primer on Deep Learning including Neural Conditional Random Fields, which appear to be taken for granted by almost all publications that use them.

In order to obtain enough data for the execution of the supervised experiments, a dataset of German novels had to be created. In chapter 5 ATHEN is presented, a general purpose annotation tool, built upon Apache UIMA. Using ATHEN, 90 segments of DROC were annotated with about 400.000 character references as well as the coreferences among them and about 4.000 direct speech utterances with their speakers and addressees. Additional datasets comprising Kindler summarizes of novels as well as a data set comprising interactions is presented alongside their guidelines in chapter 4.

The character references need to be clustered according to their coreferences in order to get a representation of the entities in a novel. Since the coreference resolution is a tough problem which has been in the center of research for half a century an extensive amount of models of the literature have been reimplemented or were used for experiments on the segments of DROC. At the current point in time, a carefully tuned rule-based algorithm can still outperform advanced techniques based on Deep Learning, but despite much effort, the current system only reached an B-Cubed F1 of about 55%. A key problem of the coreference is the design of good features, that are both reliable and not sparse. Debugging and understanding the results of a coreference is highly problematic, considering the references show very long range dependencies and mistakes ought to propagate throughout the entire text, making it hard to track the original mistake (which might have been introduced by a different preprocessing step in the first place). Experiments towards the adaptation of systems for coreference resolution are shown in chapter 8. A further issue that arises when dealing with the automatic prediction of coreferences is the evaluation scores. Even though there are plenty metrics already (see section 3.1.3 for a detailed overview) available, none of them allows for a detailed insight into which strengths and weaknesses an algorithm actually has and can therefore only be used to rank existing systems. Section 3.1.3 proposes the idea for a metric which not only builds upon the strengths of the previous metrics, while excluding their shortcomings, but also offers explainability of the results in more detailed ways as all other metrics.

The detection of speaker and addressees for direct speech utterances is an important step for the the aggregation of interactions statistics between characters. The task was split into two parts. The first part deals with the detection of speaker inside frames of direct speeches and the second parts uses the result of the first stage in order to propagate them to nearby direct speeches without a context. Rule-based approaches that base their decision on the output of the parser yield almost 94% accuracy for the first stage, and combined with a propagation algorithm can detect about 85% correct speaker among all direct speech utterances. The detection of the addressee is more complicated and could only be detected with an accuracy of about 65%, this time machine learning outperformed the rule-based approaches. The experiments are shown in chapter 7. The other main source for interactions and labelled relations between entities was examined in chapter 9. The main results are, that simple rules can extract relations that are described using genitive and possessive constructions in a very reliable way. However this does mainly include family relations, and the full use of these relations can only be seen after the coreference (the job of the coreference is to determine the entity for the corresponding references) and the current coreference has major issues with the resolution of appellatives. The detection of interactions in narrative segments and the networks that could be created using automatically extracted interactions were shown to outperform baselines that rely on co-occurence counts between entities.

The automatic extraction and evaluation of character networks is conducted in chapter 10. This chapter presents new methods for automatic evaluation that are not relying on domain experts but on summaries for a novel that can be accessed much easier compared to a manually created network. Furthermore this chapter directs the evaluation metrics from classical SNA scores more towards the evaluation of rankings between the most central entities and their edges. The evaluation showed, that information about the most central entities seems reliable, but the quality drops rather fast, with the main source of errors being the coreference resolution system.

11.2. Outlook

This work developed components for the automatic extraction of character networks, but since especially the quality of the coreference module hinders the extraction of network of high quality, it is a good idea to direct effort into the improvement of this module. Recent progress in NLP using very large language models (e.g. BERT) showed constant improvements for the English dataset. In chapter 8, the most promising Deep Learning model for automatic coreference is tested and we found, that the main issues with this component, is to "forget" important information after a while. In chapter 10, a two stage approach for the coreference was presented, which is based upon the segmentation of the novels into smaller chunks first (in the best case these chunks represent scenes). An application of a different coreference algorithm for the local segments and then recombining these results has already proved to be able to produce promising results. The coreference within these local sections could be done by a Neural network and the results could be combined using the rule-based system. This would also overcome the most critical drawback of the current Deep Learning model of requiring almost 1 Terra byte of Ram, when being executed on a larger text. The main source of error for the speaker resolution is still caused by propagated parser mistakes, so an adaptation of a parser to the domain of German literary novels would naturally improve the quality of these algorithms. To speed up the creation of gold annotations on a large number of novels, a different approach is by the usage of semi-automatic systems. The current rule-based algorithm is able to incorporate prior knowledge about the text it is processing. This prior knowledge could be provided manually in order to prevent the merge of distinct entities or help the algorithm to determine merges between appellative and names, or by providing information about the types of relations between the character. In a next step, this meta data could be derived automatically from provided summaries, therefore only a summary would have to be provided to improve the coreference algorithm.

Aside from the problem of the coreference resolution, a more fine grained description of characters could be extracted, e.g. by using the OBIE algorithm presented in section 4.9.2. An ontology is already provided by Zollner-Weber [Zöllner-Weber, 2008] and offers interesting challenges for an automatic Information Extraction system. These additional information about characters could yet again provide additional features for the coreference module.

Bibliography

[Agarwal et al., 2012] Agarwal, A., Corvalan, A., Jensen, J., and Rambow, O. (2012). Social network analysis of alice in wonderland. In *Proceedings of the NAACL-HLT 2012 Workshop on computational linguistics for literature*, pages 88–96.

[Agarwal et al., 2013] Agarwal, A., Kotalwar, A., Zheng, J., and Rambow, O. (2013). Sinnet: Social interaction network extractor from text. In *The Companion Volume of the Proceedings of IJCNLP 2013: System Demonstrations*, pages 33–36.

[Alliance, 2003] Alliance, O. (2003). *Osgi service platform, release 3*. IOS press.

[Almeida et al., 2014] Almeida, M. S., Almeida, M. B., and Martins, A. F. (2014). A joint model for quotation attribution and coreference resolution. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 39–48.

[Almugbel, 2019] Almugbel, Z. (2019). Ontology Based Approach for Annotating a Corpus of Computer Science Abstracts. In *2019 International Conference on Computer and Information Sciences (ICCIS)*, pages 1–6. IEEE.

[Anantharangachar et al., 2013] Anantharangachar, R., Ramani, S., and Rajagopalan, S. (2013). Ontology guided information extraction from unstructured text. *arXiv preprint arXiv:1302.1335*.

[Anderson, 2015] Anderson, C. (2015). Docker [software engineering]. *IEEE Software*, 32(3):102–c3.

[Andrew and Gao, 2007] Andrew, G. and Gao, J. (2007). Scalable training of L 1-regularized log-linear models. In *Proceedings of the 24th international conference on Machine learning*, pages 33–40. ACM.

[Aone and Bennett, 1995] Aone, C. and Bennett, S. W. (1995). Evaluating automated and manual acquisition of anaphora resolution strategies. In *Proceedings of the 33rd annual meeting on Association for Computational Linguistics*, pages 122–129. Association for Computational Linguistics.

[Ardanuy and Sporleder, 2014] Ardanuy, M. C. and Sporleder, C. (2014). Structure-based clustering of novels. In *Proceedings of the 3rd Workshop on Computational Linguistics for Literature (CLFL)*, pages 31–39.

[Bach and Badaskar, 2007] Bach, N. and Badaskar, S. (2007). A review of relation extraction. *Literature review for Language and Statistics II*, 2.

[Bagga and Baldwin, 1998] Bagga, A. and Baldwin, B. (1998). Algorithms for scoring coreference chains. In *The first international conference on language resources and evaluation workshop on linguistics coreference*, volume 1, pages 563–566. Granada.

[Baldridge, 2005] Baldridge, J. (2005). The opennlp project. *URL: http://opennlp. apache. org/index. html,(accessed 2 February 2012)*, page 1.

[Baldwin, 1996] Baldwin, B. (1996). Cogniac: A high precision pronoun resolution engine. In *Jill.* Citeseer.

[Bamman et al., 2014] Bamman, D., Underwood, T., and Smith, N. A. (2014). A bayesian mixed effects model of literary character. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 370–379.

[Barber, 2003] Barber, D. (2003). Probabilistic modelling and reasoning: The junction tree algorithm. *Course notes*, 2004.

[Barth et al., 2008] Barth, A., Jackson, C., Reis, C., Team, T., et al. (2008). The security architecture of the chromium browser. In *Technical report.* Stanford University.

[Baydin et al., 2018] Baydin, A. G., Pearlmutter, B. A., Radul, A. A., and Siskind, J. M. (2018). Automatic differentiation in machine learning: a survey. *Journal of machine learning research*, 18(153).

[Bengio et al., 2006] Bengio, Y., Delalleau, O., and Roux, N. L. (2006). Label propagation and quadratic criterion. In Chapelle, O., Schölkopf, B., and Zien, A., editors, *Semi-Supervised Learning*, pages 192–216. The MIT Press.

[Benikova et al., 2014] Benikova, D., Biemann, C., Kisselew, M., and Pado, S. (2014). Germeval 2014 named entity recognition shared task: companion paper.

[Berger, 1997] Berger, A. (1997). The improved iterative scaling algorithm: A gentle introduction. Technical report, Citeseer.

[Berger et al., 1996] Berger, A. L., Pietra, V. J. D., and Pietra, S. A. D. (1996). A maximum entropy approach to natural language processing. *Computational linguistics*, 22(1):39–71.

[Białek, 2010] Białek, M. (2010). Ontology-Based Information Extraction: Current Approaches. *Challenges of Modern Technology*, 1.

[Björkelund and Kuhn, 2014] Björkelund, A. and Kuhn, J. (2014). Learning structured perceptrons for coreference resolution with latent antecedents and non-local features. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 47–57.

[Blei et al., 2003] Blei, D. M., Ng, A. Y., and Jordan, M. I. (2003). Latent dirichlet allocation. *Journal of machine Learning research*, 3(Jan):993–1022.

[Bleibtreu, 1888] Bleibtreu, K. (1888). *Größenwahn. Pathologischer Roman.* Wilhelm Friedrich, Leipzig.

[Bohnet, 2010] Bohnet, B. (2010). Very high accuracy and fast dependency parsing is not a contradiction. In *Proceedings of the 23rd international conference on computational linguistics*, pages 89–97. Association for Computational Linguistics.

[Bojanowski et al., 2016] Bojanowski, P., Grave, E., Joulin, A., and Mikolov, T. (2016). Enriching Word Vectors with Subword Information. *arXiv preprint arXiv:1607.04606.*

[Boser et al., 1992] Boser, B. E., Guyon, I. M., and Vapnik, V. N. (1992). A training algorithm for optimal margin classifiers. In *Proceedings of the fifth annual workshop on Computational learning theory*, pages 144–152. ACM.

[Brand, 2002] Brand, T. (2002). *Theodor Fontane „Effi Briest". In: Königs Erläuterungen und Materialien.* C. Bange Verlag.

[Brants et al., 2002] Brants, S., Dipper, S., Hansen, S., Lezius, W., and Smith, G. (2002). The TIGER treebank. In *Proceedings of the workshop on treebanks and linguistic theories*, volume 168.

[Brennan et al., 1987] Brennan, S. E., Friedman, M. W., and Pollard, C. J. (1987). A centering approach to pronouns. In *Proceedings of the 25th annual meeting on Association for Computational Linguistics*, pages 155–162. Association for Computational Linguistics.

[Brown et al., 1992a] Brown, P. F., Desouza, P. V., Mercer, R. L., Pietra, V. J. D., and Lai, J. C. (1992a). Class-based n-gram models of natural language. *Computational linguistics*, 18(4):467–479.

[Brown et al., 1992b] Brown, P. F., Desouza, P. V., Mercer, R. L., Pietra, V. J. D., and Lai, J. C. (1992b). Class-based n-gram models of natural language. *Computational linguistics*, 18(4):467–479.

[Brunner et al., 2019] Brunner, A., Weimer, L., Tu, N. D. T., Engelberg, S., and Jannidis, F. (2019). Das Redewiedergabe-Korpus. Eine neue Ressource. *Proceedings of Dhd.*

[Buitelaar et al., 2006] Buitelaar, P., Cimiano, P., Racioppa, S., and Siegel, M. (2006). Ontology-based information extraction with soba. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC).*

[Carter, 1986] Carter, D. M. (1986). *A shallow processing approach to anaphor resolution.* PhD thesis, University of Cambridge.

[Celikyilmaz et al., 2010] Celikyilmaz, A., Hakkani-Tur, D., He, H., Kondrak, G., and Barbosa, D. (2010). The actortopic model for extracting social networks in literary narrative. In *NIPS Workshop: Machine Learning for Social Computing.*

[Chaganty and Muzny, 2014] Chaganty, A. and Muzny, G. (2014). Quote attribution for literary text with neural networks. *Avaliable at http://cs224d. stanford. edu/reports/ChagantyArun. pdf.*

[Chambers et al., 2010] Chambers, A., Smyth, P., and Steyvers, M. (2010). Learning concept graphs from text with stick-breaking priors. In *Advances in Neural Information Processing Systems*, pages 334–342.

[Chang and Lin, 2011] Chang, C.-C. and Lin, C.-J. (2011). LIBSVM: A library for support vector machines. *ACM transactions on intelligent systems and technology (TIST)*, 2(3):27.

[Chawla et al., 2004] Chawla, N. V., Japkowicz, N., and Kotcz, A. (2004). Special issue on learning from imbalanced data sets. *ACM Sigkdd Explorations Newsletter*, 6(1):1–6.

[Chen and Manning, 2014] Chen, D. and Manning, C. (2014). A fast and accurate dependency parser using neural networks. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 740–750.

[Chinchor and Robinson, 1997] Chinchor, N. and Robinson, P. (1997). MUC-7 named entity task definition. In *Proceedings of the 7th Conference on Message Understanding*, volume 29.

[Cho et al., 2014] Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. (2014). Learning phrase representations using RNN encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078.*

[Cimiano et al., 2005] Cimiano, P., Hotho, A., and Staab, S. (2005). Learning concept hierarchies from text corpora using formal concept analysis. *Journal of artificial intelligence research*, 24:305–339.

[Clark and Manning, 2016a] Clark, K. and Manning, C. D. (2016a). Deep Reinforcement Learning for Mention-Ranking Coreference Models. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2256–2262.

[Clark and Manning, 2016b] Clark, K. and Manning, C. D. (2016b). Improving Coreference Resolution by Learning Entity-Level Distributed Representations. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 643–653.

[Clarke et al., 2009] Clarke, J., Connors, J., and Bruno, E. J. (2009). *JavaFX: Developing Rich Internet Applications.* Pearson Education.

[Cohen, 1960] Cohen, J. (1960). A coefficient of agreement for nominal scales. *Educational and psychological measurement*, 20(1):37–46.

[Collins, 2002] Collins, M. (2002). Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *Proceedings of the*

ACL-02 conference on Empirical methods in natural language processing-Volume 10, pages 1–8. Association for Computational Linguistics.

[Collins, 2015] Collins, M. (2015). Log-linear models, memms, and crfs.

[Collobert et al., 2011] Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K., and Kuksa, P. (2011). Natural language processing (almost) from scratch. *Journal of machine learning research*, 12(Aug):2493–2537.

[Connolly et al., 1997] Connolly, D., Burger, J. D., and Day, D. S. (1997). A machine learning approach to anaphoric reference. In *New methods in language processing*, pages 133–144.

[Crammer et al., 2006] Crammer, K., Dekel, O., Keshet, J., Shalev-Shwartz, S., and Singer, Y. (2006). Online passive-aggressive algorithms. *Journal of Machine Learning Research*, 7(Mar):551–585.

[Crammer and Singer, 2001] Crammer, K. and Singer, Y. (2001). On the algorithmic implementation of multiclass kernel-based vector machines. *Journal of machine learning research*, 2(Dec):265–292.

[Crammer and Singer, 2003] Crammer, K. and Singer, Y. (2003). Ultraconservative online algorithms for multiclass problems. *Journal of Machine Learning Research*, 3(Jan):951–991.

[Craven et al., 1999] Craven, M., Kumlien, J., et al. (1999). Constructing biological knowledge bases by extracting information from text sources. In *ISMB*, volume 1999, pages 77–86.

[Cunningham et al., 1999] Cunningham, H., Maynard, D., and Tablan, V. (1999). JAPE: a Java annotation patterns engine.

[Cunningham et al., 2013] Cunningham, H., Tablan, V., Roberts, A., and Bontcheva, K. (2013). Getting more out of biomedical documents with GATE's full lifecycle open source text analytics. *PLoS computational biology*, 9(2):e1002854.

[Darroch and Ratcliff, 1972] Darroch, J. N. and Ratcliff, D. (1972). Generalized iterative scaling for log-linear models. *The annals of mathematical statistics*, pages 1470–1480.

[Daumé III, 2012] Daumé III, H. (2012). A course in machine learning. *Publisher, ciml. info*, 5:69.

[Dauthendey, 1923] Dauthendey, M. (1923). *Lingam. Zwölf asiatische Novellen.* Albert Langen, München.

[de Castilho et al., 2016] de Castilho, R. E., Mujdricza-Maydt, E., Yimam, S. M., Hartmann, S., Gurevych, I., Frank, A., and Biemann, C. (2016). A web-based tool for the integrated annotation of semantic and syntactic structures. In *Proceedings of* *the Workshop on Language Technology Resources and Tools for Digital Humanities (LT4DH)*, pages 76–84.

[Decoste and Schölkopf, 2002] Decoste, D. and Schölkopf, B. (2002). Training invariant support vector machines. *Machine learning*, 46(1-3):161–190.

[Dekker et al., 2018] Dekker, N., Kuhn, T., and van Erp, M. (2018). Evaluating social network extraction for classic and modern fiction literature. *PeerJ PrePrints*, 6:e27263v1.

[Devlin et al., 2018a] Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2018a). Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

[Devlin et al., 2018b] Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2018b). Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

[Didakowski et al., 2007] Didakowski, J., Geyken, A., and Hanneforth, T. (2007). Eigennamenerkennung zwischen morphologischer Analyse und Part-of-Speech Tagging: ein automatentheoriebasierter Ansatz. *Zeitschrift für Sprachwissenschaft*, 26(2):157–186.

[Dijkstra, 1959] Dijkstra, E. W. (1959). A note on two problems in connexion with graphs. *Numerische mathematik*, 1(1):269–271.

[Doddington et al., 2004] Doddington, G. R., Mitchell, A., Przybocki, M. A., Ramshaw, L. A., Strassel, S. M., and Weischedel, R. M. (2004). The Automatic Content Extraction (ACE) Program-Tasks, Data, and Evaluation. In *Lrec*, volume 2, page 1. Lisbon.

[Dohm, 1894] Dohm, H. (1894). *Wie Frauen werden*. Schlesische Buchdruckerei, Kunst- und Verlags-Anstalt v. S. Schottlaender, Breslau.

[Druck et al., 2008] Druck, G., Mann, G., and McCallum, A. (2008). Learning from labeled features using generalized expectation criteria. In *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 595–602. ACM.

[Drumond and Girardi, 2008] Drumond, L. and Girardi, R. (2008). A Survey of Ontology Learning Procedures. *WONTO*, 427:1–13.

[Edwards et al., 2018] Edwards, M., Mitchell, L., Tuke, J., and Roughan, M. (2018). The one comparing narrative social network extraction techniques. *arXiv preprint arXiv:1811.01467*.

[Elson et al., 2010] Elson, D. K., Dames, N., and McKeown, K. R. (2010). Extracting social networks from literary fiction. In *Proceedings of the 48th annual meeting of the association for computational linguistics*, pages 138–147. Association for Computational Linguistics.

[Elson and McKeown, 2010] Elson, D. K. and McKeown, K. R. (2010). Automatic attribution of quoted speech in literary narrative. In *Twenty-Fourth AAAI Conference on Artificial Intelligence*.

[Ephraim and Merhav, 2002] Ephraim, Y. and Merhav, N. (2002). Hidden markov processes. *IEEE Transactions on information theory*, 48(6):1518–1569.

[Faruqui et al., 2010] Faruqui, M., Padó, S., and Sprachverarbeitung, M. (2010). Training and Evaluating a German Named Entity Recognizer with Semantic Generalization. In *KONVENS*, pages 129–133.

[Fernandes et al., 2014] Fernandes, E. R., dos Santos, C. N., and Milidiú, R. L. (2014). Latent trees for coreference resolution. *Computational Linguistics*, 40(4):801–835.

[Finkel et al., 2005] Finkel, J. R., Grenager, T., and Manning, C. (2005). Incorporating non-local information into information extraction systems by gibbs sampling. In *Proceedings of the 43rd annual meeting on association for computational linguistics*, pages 363–370. Association for Computational Linguistics.

[Finn and Kushmerick, 2003] Finn, A. and Kushmerick, N. (2003). Active learning selection strategies for information extraction. In *Proceedings of the International Workshop on Adaptive Text Extraction and Mining (ATEM-03)*, pages 18–25.

[Fischer, 2017] Fischer, E. (2017). Automatische extraktion von interaktionen zwischen zwei personen in literarischen texten. Master's thesis, University of Wuerzburg.

[Forney, 1973] Forney, G. D. (1973). The viterbi algorithm. *Proceedings of the IEEE*, 61(3):268–278.

[Foth, 2006] Foth, K. A. (2006). Eine umfassende Constraint-Dependenz-Grammatik des Deutschen.

[Friedman et al., 1997] Friedman, N., Geiger, D., and Goldszmidt, M. (1997). Bayesian network classifiers. *Machine learning*, 29(2-3):131–163.

[Gábor et al., 2018] Gábor, K., Buscaldi, D., Schumann, A.-K., QasemiZadeh, B., Zargayouna, H., and Charnois, T. (2018). Semeval-2018 Task 7: Semantic relation extraction and classification in scientific papers. In *Proceedings of The 12th International Workshop on Semantic Evaluation*, pages 679–688.

[Gédéon, 2010] Gédéon, W. J. (2010). *OSGi and Apache Felix 3.0 Beginner's Guide*. Packt Publishing Ltd.

[Gennari et al., 2003] Gennari, J. H., Musen, M. A., Fergerson, R. W., Grosso, W. E., Crubézy, M., Eriksson, H., Noy, N. F., and Tu, S. W. (2003). The evolution of Protégé: an environment for knowledge-based systems development. *International Journal of Human-computer studies*, 58(1):89–123.

[Gerdes et al., 2013] Gerdes, K. et al. (2013). Predictive incremental parsing and its evaluation. *Computational Dependency Theory*, 258:186.

[Glass and Bangay, 2006] Glass, K. and Bangay, S. (2006). Hierarchical rule generalisation for speaker identification in fiction books. In *Proceedings of the 2006 annual research conference of the South African institute of computer scientists and information technologists on IT research in developing countries*, pages 31–40. South African Institute for Computer Scientists and Information Technologists.

[Glass and Bangay, 2007] Glass, K. and Bangay, S. (2007). A naive salience-based method for speaker identification in fiction books. In *Proceedings of the 18th Annual Symposium of the Pattern Recognition Association of South Africa (PRASA'07)*, pages 1–6.

[Goldberg, 2017] Goldberg, Y. (2017). Neural network methods for natural language processing. *Synthesis Lectures on Human Language Technologies*, 10(1):1–309.

[Goodfellow et al., 2016] Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep learning*. MIT press.

[Gotz and Suhre, 2004] Gotz, T. and Suhre, O. (2004). Design and implementation of the UIMA Common Analysis System. *IBM Systems Journal*, 43(3):476–489.

[Graves et al., 2013] Graves, A., Mohamed, A.-r., and Hinton, G. (2013). Speech recognition with deep recurrent neural networks. In *2013 IEEE international conference on acoustics, speech and signal processing*, pages 6645–6649. IEEE.

[Grishman and Sundheim, 1996] Grishman, R. and Sundheim, B. (1996). Message Understanding Conference-6: A Brief History. In *Coling*, volume 96, pages 466–471.

[Hammerton, 2003] Hammerton, J. (2003). Named entity recognition with long short-term memory. In *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003-Volume 4*, pages 172–175. Association for Computational Linguistics.

[Hamp and Feldweg, 1997] Hamp, B. and Feldweg, H. (1997). Germanet-a lexical-semantic net for german. *Automatic Information Extraction and Building of Lexical Semantic Resources for NLP Applications.*

[Hassan et al., 2012] Hassan, A., Abu-Jbara, A., and Radev, D. (2012). Extracting signed social networks from text. In *Workshop Proceedings of TextGraphs-7 on Graph-based Methods for Natural Language Processing*, pages 6–14. Association for Computational Linguistics.

[He et al., 2013] He, H., Barbosa, D., and Kondrak, G. (2013). Identification of speakers in novels. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1312–1320.

[Hearst, 1992] Hearst, M. A. (1992). Automatic acquisition of hyponyms from large text corpora. In *Proceedings of the 14th conference on Computational linguistics-Volume 2*, pages 539–545. Association for Computational Linguistics.

[Hestenes and Stiefel, 1952] Hestenes, M. R. and Stiefel, E. (1952). *Methods of conjugate gradients for solving linear systems*, volume 49. NBS Washington, DC.

[Hobbs, 1978] Hobbs, J. R. (1978). Resolving pronoun references. *Lingua*, 44(4):311–338.

[Hochreiter and Schmidhuber, 1997] Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9(8):1735–1780.

[Huang et al., 2015] Huang, Z., Xu, W., and Yu, K. (2015). Bidirectional LSTM-CRF models for sequence tagging. *arXiv preprint arXiv:1508.01991.*

[Iida et al., 2003] Iida, R., Inui, K., Takamura, H., and Matsumoto, Y. (2003). Incorporating contextual cues in trainable models for coreference resolution. In *Proceedings of the EACL Workshop on The Computational Treatment of Anaphora*, pages 23–30. Citeseer.

[Iosif and Mishra, 2014] Iosif, E. and Mishra, T. (2014). From Speaker Identification to Affective Analysis: A Multi-Step System for Analyzing Children's Stories. In *Proceedings of the 3rd Workshop on Computational Linguistics for Literature (CLFL)*, pages 40–49.

[Jannidis et al., 2018] Jannidis, F., Konle, L., Zehe, A., Hotho, A., and Krug, M. (2018). Analysing direct speech in german novels. In *DHd 2018.*

[Jannidis et al., 2017] Jannidis, F., Reger, I., Weimer, L., Krug, M., Toepfer, M., and Puppe, F. (2017). Automatische Erkennung von Figuren in deutschsprachigen Romanen. *Proceedings of DhD.*

[Jayannavar et al., 2015] Jayannavar, P., Agarwal, A., Ju, M., and Rambow, O. (2015). Validating literary theories using automatic social network extraction. In *Proceedings of the Fourth Workshop on Computational Linguistics for Literature*, pages 32–41.

[Jaynes, 1957] Jaynes, E. T. (1957). Information theory and statistical mechanics. *Physical review*, 106(4):620.

[Jiang and Zhai, 2007] Jiang, J. and Zhai, C. (2007). A systematic exploration of the feature space for relation extraction. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Proceedings of the Main Conference*, pages 113–120.

[Jing et al., 2007] Jing, H., Kambhatla, N., and Roukos, S. (2007). Extracting social networks and biographical facts from conversational speech transcripts. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 1040–1047.

[Joachims, 1999] Joachims, T. (1999). Transductive inference for text classification using support vector machines. In *Icml*, volume 99, pages 200–209.

[Joulin et al., 2016] Joulin, A., Grave, E., Bojanowski, P., and Mikolov, T. (2016). Bag of tricks for efficient text classification. *arXiv preprint arXiv:1607.01759*.

[Jung et al., 2012] Jung, H., Choi, S.-P., Lee, S., and Song, S.-K. (2012). *Survey on Kernel-Based Relation Extraction*. InTech.

[Kabadjov, 2007] Kabadjov, M. (2007). *Task-oriented evaluation of anaphora resolution*. PhD thesis, Ph. D. thesis, University of Essex, Colchester, UK.

[Kambhatla, 2004] Kambhatla, N. (2004). Combining lexical, syntactic, and semantic features with maximum entropy models for extracting relations. In *Proceedings of the ACL 2004 on Interactive poster and demonstration sessions*, page 22. Association for Computational Linguistics.

[Kasami, 1966] Kasami, T. (1966). An efficient recognition and syntax-analysis algorithm for context-free languages. *Coordinated Science Laboratory Report no. R-257*.

[Kingma and Ba, 2014] Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

[Kiss and Strunk, 2006] Kiss, T. and Strunk, J. (2006). Unsupervised multilingual sentence boundary detection. *Computational Linguistics*, 32(4):485–525.

[Klinger and Tomanek, 2007] Klinger, R. and Tomanek, K. (2007). *Classical probabilistic models and conditional random fields*. Citeseer.

[Kluegl et al., 2016] Kluegl, P., Toepfer, M., Beck, P.-D., Fette, G., and Puppe, F. (2016). UIMA Ruta: Rapid development of rule-based information extraction applications. *Natural Language Engineering*, 22(1):1–40.

[KONYS, 2015] KONYS, A. (2015). An Approach for Ontology-Based Information Extraction System Selection and Evaluation. *Przegląd Elektrotechniczny*, 91(11):205–209.

[Krug et al., 2016] Krug, M., Jannidis, F., Reger, I., Weimer, L., Macharowsky, L., and Puppe, F. (2016). Attribuierung direkter Reden in deutschen Romanen des 18.-20. Jahrhunderts: Methoden zur Bestimmung des Sprechers und des Angesprochenen. DHd Tagung.

[Krug et al., 2015] Krug, M., Puppe, F., Jannidis, F., Macharowsky, L., Reger, I., and Weimar, L. (2015). Rule-based coreference resolution in German historic novels. In *Proceedings of the Fourth Workshop on Computational Linguistics for Literature*, pages 98–104.

[Krug et al., 2018a] Krug, M., Puppe, F., Reger, I., Weimer, L., Macharowsky, L., Feldhaus, S., and Jannidis, F. (2018a). Description of a Corpus of Character References in German Novels - DROC [Deutsches ROman Corpus]. In *DARIAH-DE Working Papers*. DARIAH-DE.

[Krug et al., 2017a] Krug, M., Reger, I., Jannidis, F., Weimer, L., Madarász, N., and Puppe, F. (2017a). Overcoming Data Sparsity for Relation Detection in German Novels. In *DH*.

[Krug et al., 2019a] Krug, M., Schmidt, D., Jannidis, F., and Puppe, F. (2019a). Techniques for High Quality Character Reference Detection on German Historical Novels. Manuscript submitted for publication at De Gruyter Open Linguistics.

[Krug et al., 2019b] Krug, M., Schmidt, D., Wehner, N., Jannidis, F., and Puppe, F. (2019b). Evaluation of state of the art methods for coreference resolution and quotation attribution on German literary novels. Manuscript submitted for publication at Journal of Natural Language Engineering.

[Krug et al., 2018b] Krug, M., Tu, N. D. T., Weimer, L., Reger, I., Konle, L., Jannidis, F., and Puppe, F. (2018b). Annotation and beyond – using ATHEN annotation and text highlighting environment. In *DHd 2018*.

[Krug et al., 2017b] Krug, M., Wick, C., Jannidis, F., Reger, I., Weimer, L., Madarasz, N., and Puppe, F. (2017b). Comparison of methods for automatic relation extraction in german novels. *Dhd Tagung*.

[Kschischang et al., 2001] Kschischang, F. R., Frey, B. J., Loeliger, H.-A., et al. (2001). Factor graphs and the sum-product algorithm. *IEEE Transactions on information theory*, 47(2):498–519.

[Kübler et al., 2009] Kübler, S., McDonald, R., and Nivre, J. (2009). Dependency parsing. *Synthesis Lectures on Human Language Technologies*, 1(1):1–127.

[Kuhn, 1955] Kuhn, H. W. (1955). The Hungarian method for the assignment problem. *Naval research logistics quarterly*, 2(1-2):83–97.

[Lafferty et al., 2001] Lafferty, J., McCallum, A., and Pereira, F. C. (2001). Conditional random fields: Probabilistic models for segmenting and labeling sequence data.

[Lappin and Leass, 1994] Lappin, S. and Leass, H. J. (1994). An algorithm for pronominal anaphora resolution. *Computational linguistics*, 20(4):535–561.

[LeCun et al., 1998] LeCun, Y., Bottou, L., Bengio, Y., Haffner, P., et al. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.

[Lee et al., 2013] Lee, H., Chang, A., Peirsman, Y., Chambers, N., Surdeanu, M., and Jurafsky, D. (2013). Deterministic coreference resolution based on entity-centric, precision-ranked rules. *Computational Linguistics*, 39(4):885–916.

[Lee et al., 2011] Lee, H., Peirsman, Y., Chang, A., Chambers, N., Surdeanu, M., and Jurafsky, D. (2011). Stanford's multi-pass sieve coreference resolution system at the CoNLL-2011 shared task. In *Proceedings of the fifteenth conference on computational natural language learning: Shared task*, pages 28–34. Association for Computational Linguistics.

[Lee and Wong, 2016] Lee, J. and Wong, T.-s. (2016). Conversational Network in the Chinese Buddhist Canon. *Open Linguistics*, 2(1).

[Lee and Yeung, 2012] Lee, J. and Yeung, C. Y. (2012). Extracting networks of people and places from literary texts. In *Proceedings of the 26th Pacific Asia Conference on Language, Information, and Computation*, pages 209–218.

[Lee et al., 2017] Lee, K., He, L., Lewis, M., and Zettlemoyer, L. (2017). End-to-end Neural Coreference Resolution. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 188–197.

[Lehmann and Hitzler, 2007] Lehmann, J. and Hitzler, P. (2007). A Refinement Operator Based Learning Algorithm for the ALC Description Logic. In *International Conference on Inductive Logic Programming*, pages 147–160. Springer.

[Lezius, 2000] Lezius, W. (2000). Morphy-German morphology, part-of-speech tagging and applications. In *Proceedings of the 9th EURALEX International Congress*, pages 619–623. University of Stuttgart Stuttgart.

[Lloyd, 1982] Lloyd, S. (1982). Least squares quantization in PCM. *IEEE transactions on information theory*, 28(2):129–137.

[Loy et al., 2002] Loy, M., Eckstein, R., Wood, D., Elliott, J., and Cole, B. (2002). *Java swing*. " O'Reilly Media, Inc.".

[Lozano-Perez, 2001] Lozano-Perez, T. (2001). Sequential minimal optimization for svm.

[Luo, 2005] Luo, X. (2005). On coreference resolution performance metrics. In *Proceedings of the conference on human language technology and empirical methods in natural language processing*, pages 25–32. Association for Computational Linguistics.

[Luo et al., 2014] Luo, X., Pradhan, S., Recasens, M., and Hovy, E. (2014). An extension of BLANC to system mentions. In *Proceedings of the conference. Association for Computational Linguisticse Meeting*, volume 2014, page 24. NIH Public Access.

[Maedche and Staab, 2000] Maedche, A. and Staab, S. (2000). The text-to-onto ontology learning environment. In *Software Demonstration at ICCS-2000-Eight International Conference on Conceptual Structures*, volume 38, pages 890–930. sn.

[Maedche and Staab, 2001] Maedche, A. and Staab, S. (2001). Ontology learning for the semantic web. *IEEE Intelligent systems*, 16(2):72–79.

[Manning et al., 2014] Manning, C., Surdeanu, M., Bauer, J., Finkel, J., Bethard, S., and McClosky, D. (2014). The Stanford CoreNLP natural language processing toolkit. In *Proceedings of 52nd annual meeting of the association for computational linguistics: system demonstrations*, pages 55–60.

[Massey et al., 2015] Massey, P., Xia, P., Bamman, D., and Smith, N. A. (2015). Annotating character relationships in literary texts. *arXiv preprint arXiv:1512.00728*.

[McAffer et al., 2010] McAffer, J., Lemieux, J.-M., and Aniszczyk, C. (2010). *Eclipse rich client platform*. Addison-Wesley Professional.

[McCallum et al., 2000] McCallum, A., Freitag, D., and Pereira, F. C. (2000). Maximum Entropy Markov Models for Information Extraction and Segmentation. In *Icml*, volume 17, pages 591–598.

[McCallum, 2002] McCallum, A. K. (2002). Mallet: A machine learning for language toolkit.

[McCandless et al., 2010] McCandless, M., Hatcher, E., and Gospodnetic, O. (2010). *Lucene in action: covers Apache Lucene 3.0*. Manning Publications Co.

[McCann et al., 2017] McCann, B., Bradbury, J., Xiong, C., and Socher, R. (2017). Learned in translation: Contextualized word vectors. In *Advances in Neural Information Processing Systems*, pages 6294–6305.

[McCarthy and Lehnert, 1995] McCarthy, J. F. and Lehnert, W. G. (1995). Using decision trees for coreference resolution. *arXiv preprint cmp-lg/9505043*.

[McDonald et al., 2006] McDonald, R., Lerman, K., and Pereira, F. (2006). Multilingual dependency analysis with a two-stage discriminative parser. In *Proceedings of the Tenth Conference on Computational Natural Language Learning*, pages 216–220. Association for Computational Linguistics.

[McGuinness et al., 2004] McGuinness, D. L., Van Harmelen, F., et al. (2004). OWL web ontology language overview. *W3C recommendation*, 10(10):2004.

[Meister et al., 2017] Meister, J. C., Gius, E., Horstmann, J., Jacke, J., and Petris, M. (2017). CATMA 5.0 Tutorial. In *DH*.

[Merks et al., 2003] Merks, E., Eliersick, R., Grose, T., Budinsky, F., and Steinberg, D. (2003). The eclipse modeling framework. *retrieved from, total*, page 37.

[Mikolov et al., 2013] Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., and Dean, J. (2013). Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

[Miller et al., 2010] Miller, F. P., Vandome, A. F., and McBrewster, J. (2010). Apache Maven.

[Miller, 1998] Miller, G. (1998). *WordNet: An electronic lexical database.* MIT press.

[Miller, 1995] Miller, G. A. (1995). WordNet: a lexical database for English. *Communications of the ACM*, 38(11):39–41.

[Minh et al., 2006] Minh, H. Q., Niyogi, P., and Yao, Y. (2006). Mercer's theorem, feature maps, and smoothing. In *International Conference on Computational Learning Theory*, pages 154–168. Springer.

[Mintz et al., 2009] Mintz, M., Bills, S., Snow, R., and Jurafsky, D. (2009). Distant supervision for relation extraction without labeled data. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2-Volume 2*, pages 1003–1011. Association for Computational Linguistics.

[Mitkov, 1998] Mitkov, R. (1998). Robust pronoun resolution with limited knowledge. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics-Volume 2*, pages 869–875. Association for Computational Linguistics.

[Mitkov et al., 2002] Mitkov, R., Evans, R., and Orasan, C. (2002). A new, fully automatic version of Mitkov's knowledge-poor pronoun resolution method. In *International Conference on Intelligent Text Processing and Computational Linguistics*, pages 168–186. Springer.

[Molina et al., 2002] Molina, L. C., Belanche, L., and Nebot, À. (2002). Feature selection algorithms: A survey and experimental evaluation. In *2002 IEEE International Conference on Data Mining, 2002. Proceedings.*, pages 306–313. IEEE.

[Moosavi and Strube, 2016] Moosavi, N. S. and Strube, M. (2016). Which coreference evaluation metric do you trust? a proposal for a link-based entity aware metric. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 632–642.

[Morton and LaCivita, 2003] Morton, T. and LaCivita, J. (2003). WordFreak: an open tool for linguistic annotation. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology: Demonstrations-Volume 4*, pages 17–18. Association for Computational Linguistics.

[Müller and Strube, 2006] Müller, C. and Strube, M. (2006). Multi-level annotation of linguistic data with MMAX2. *Corpus technology and language pedagogy: New resources, new tools, new methods*, 3:197–214.

[Muzny et al., 2017] Muzny, G., Fang, M., Chang, A., and Jurafsky, D. (2017). A two-stage sieve approach for quote attribution. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 460–470.

[Neudecker, 2016] Neudecker, C. (2016). An open corpus for named entity recognition in historic newspapers. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*, pages 4348–4352.

[Neves and Leser, 2012] Neves, M. and Leser, U. (2012). A survey on annotation tools for the biomedical literature. *Briefings in bioinformatics*, 15(2):327–340.

[Nivre, 2003] Nivre, J. (2003). An efficient algorithm for projective dependency parsing. In *Proceedings of the Eighth International Conference on Parsing Technologies*, pages 149–160.

[Nocedal, 1980] Nocedal, J. (1980). Updating quasi-Newton matrices with limited storage. *Mathematics of computation*, 35(151):773–782.

[Nvidia, 2019] Nvidia (2019). Megatronlm: Training billion+ parameter language models using gpu model parallelism.

[Ogren, 2006] Ogren, P. V. (2006). Knowtator: a protégé plug-in for annotated corpus construction. In *Proceedings of the 2006 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology: companion volume: demonstrations*, pages 273–275. Association for Computational Linguistics.

[O'Keefe et al., 2012] O'Keefe, T., Pareti, S., Curran, J. R., Koprinska, I., and Honnibal, M. (2012). A sequence labelling approach to quote attribution. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 790–799. Association for Computational Linguistics.

[O'Donnell, 2008] O'Donnell, M. (2008). Demonstration of the UAM CorpusTool for text and image annotation. In *Proceedings of the ACL-08: HLT Demo Session*, pages 13–16.

[Park et al., 2013] Park, G.-M., Kim, S.-H., Hwang, H.-R., and Cho, H.-G. (2013). Complex system analysis of social networks extracted from literary fictions. *International Journal of Machine Learning and Computing*, 3(1):107.

[Pearl and Mackenzie, 2018] Pearl, J. and Mackenzie, D. (2018). *The book of why: the new science of cause and effect.* Basic Books.

[Pennington et al., 2014] Pennington, J., Socher, R., and Manning, C. (2014). Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

[Peters et al., 2018] Peters, M. E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., and Zettlemoyer, L. (2018). Deep contextualized word representations. *arXiv preprint arXiv:1802.05365.*

[Picard et al., 2013] Picard, D., Thome, N., and Cord, M. (2013). JKernelMachines: a simple framework for kernel machine. *Journal of Machine Learning Research*, 14(1):1417–1421.

[Platt, 1998] Platt, J. (1998). Sequential minimal optimization: A fast algorithm for training support vector machines.

[Poesio et al., 2016] Poesio, M., Stuckardt, R., and Versley, Y. (2016). *Anaphora Resolution*. Springer.

[Popov et al., 2004] Popov, B., Kiryakov, A., Ognyanoff, D., Manov, D., and Kirilov, A. (2004). KIM–a semantic platform for information extraction and retrieval. *Natural language engineering*, 10(3-4):375–392.

[Pradhan et al., 2012] Pradhan, S., Moschitti, A., Xue, N., Uryupina, O., and Zhang, Y. (2012). CoNLL-2012 shared task: Modeling multilingual unrestricted coreference in OntoNotes. In *Joint Conference on EMNLP and CoNLL-Shared Task*, pages 1–40. Association for Computational Linguistics.

[Rabiner, 1989] Rabiner, L. R. (1989). A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286.

[Radford et al., 2019] Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., and Sutskever, I. (2019). Language models are unsupervised multitask learners. *OpenAI Blog*, 1(8).

[Raghunathan et al., 2010] Raghunathan, K., Lee, H., Rangarajan, S., Chambers, N., Surdeanu, M., Jurafsky, D., and Manning, C. (2010). A multi-pass sieve for coreference resolution. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 492–501. Association for Computational Linguistics.

[Rahman and Ng, 2009] Rahman, A. and Ng, V. (2009). Supervised models for coreference resolution. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 2-Volume 2*, pages 968–977. Association for Computational Linguistics.

[Rand, 1971] Rand, W. M. (1971). Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical association*, 66(336):846–850.

[Ratner et al., 2017] Ratner, A., Bach, S. H., Ehrenberg, H., Fries, J., Wu, S., and Ré, C. (2017). Snorkel: Rapid training data creation with weak supervision. *Proceedings of the VLDB Endowment*, 11(3):269–282.

[Ratner et al., 2016] Ratner, A. J., De Sa, C. M., Wu, S., Selsam, D., and Ré, C. (2016). Data programming: Creating large training sets, quickly. In *Advances in neural information processing systems*, pages 3567–3575.

[Recasens and Hovy, 2011] Recasens, M. and Hovy, E. (2011). BLANC: Implementing the Rand index for coreference evaluation. *Natural Language Engineering*, 17(4):485–510.

[Reger, 2017] Reger, I. (2017). Figurennetzwerke als Ähnlichkeitsmaß.

[Riedl and Padó, 2018] Riedl, M. and Padó, S. (2018). A named entity recognition shootout for German. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 120–125.

[Rizvi et al., 2018] Rizvi, S. T. R., Mercier, D., Agne, S., Erkel, S., Dengel, A., and Ahmed, S. (2018). Ontology-based Information Extraction from Technical Documents. In *ICAART (2)*, pages 493–500.

[Robbins and Monro, 1951] Robbins, H. and Monro, S. (1951). A stochastic approximation method. *The annals of mathematical statistics*, pages 400–407.

[Rockafellar, 1993] Rockafellar, R. T. (1993). Lagrange multipliers and optimality. *SIAM review*, 35(2):183–238.

[Rosenblatt, 1958] Rosenblatt, F. (1958). The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386.

[Rösiger and Kuhn, 2016] Rösiger, I. and Kuhn, J. (2016). IMS HotCoref DE: A Data-driven Co-reference Resolver for German. In *LREC*.

[Rumelhart et al., 1988] Rumelhart, D. E., Hinton, G. E., Williams, R. J., et al. (1988). Learning representations by back-propagating errors. *Cognitive modeling*, 5(3):1.

[Ruppenhofer et al., 2010] Ruppenhofer, J., Sporleder, C., and Shirokov, F. (2010). Speaker Attribution in Cabinet Protocols. In *Proceedings of the Seventh conference on International Language Resources and Evaluation (LREC'10)*.

[Saggion et al., 2007] Saggion, H., Funk, A., Maynard, D., and Bontcheva, K. (2007). Ontology-based information extraction for business intelligence. In *The Semantic Web*, pages 843–856. Springer.

[Sang and De Meulder, 2003] Sang, E. F. and De Meulder, F. (2003). Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. *arXiv preprint cs/0306050*.

[Sarawagi and Cohen, 2005] Sarawagi, S. and Cohen, W. W. (2005). Semi-markov conditional random fields for information extraction. In *Advances in neural information processing systems*, pages 1185–1192.

[Sassano and Utsuro, 2000] Sassano, M. and Utsuro, T. (2000). Named entity chunking techniques in supervised learning for Japanese named entity recognition. In *Proceedings of the 18th conference on Computational linguistics-Volume 2*, pages 705–711. Association for Computational Linguistics.

[Scheible et al., 2016] Scheible, C., Klinger, R., and Padó, S. (2016). Model architectures for quotation detection. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1736–1745.

[Schmid, 1995] Schmid, H. (1995). Treetagger| a language independent part-of-speech tagger. *Institut für Maschinelle Sprachverarbeitung, Universität Stuttgart*, 43:28.

[Schmid, 2005] Schmid, H. (2005). A programming language for finite state transducers. In *FSMNLP*, volume 4002, pages 308–309.

[Schmid et al., 2004] Schmid, H., Fitschen, A., and Heid, U. (2004). SMOR: A German Computational Morphology Covering Derivation, Composition and Inflection. In *LREC*, pages 1–263. Lisbon.

[Schmid and Laws, 2008] Schmid, H. and Laws, F. (2008). Estimation of conditional probabilities with decision trees and an application to fine-grained POS tagging. In *Proceedings of the 22nd International Conference on Computational Linguistics-Volume 1*, pages 777–784. Association for Computational Linguistics.

[Schmidt, 2016] Schmidt, H.-G. (2016). Kallimachos: Digital humanities als auftrag der universitätsbibliothek würzburg. *ABI Technik*, 36(3):170–177.

[Schneider, 2008] Schneider, G. (2008). *Hybrid long-distance functional dependency parsing.* PhD thesis, University of Zurich.

[Schöch et al., 2016] Schöch, C., Schlör, D., Popp, S., Brunner, A., Henny, U., and Tello, J. C. (2016). Straight Talk! Automatic Recognition of Direct Speech in Nineteenth-Century French Novels. In *DH*, pages 346–353.

[Sennrich et al., 2009] Sennrich, R., Schneider, G., Volk, M., and Warin, M. (2009). A new hybrid dependency parser for German. *Proceedings of the German Society for Computational Linguistics and Language Technology*, 115:124.

[Settles, 2009] Settles, B. (2009). Active learning literature survey. Technical report, University of Wisconsin-Madison Department of Computer Sciences.

[Sha and Pereira, 2003] Sha, F. and Pereira, F. (2003). Shallow parsing with conditional random fields. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*, pages 134–141. Association for Computational Linguistics.

[Shalev-Shwartz et al., 2011] Shalev-Shwartz, S., Singer, Y., Srebro, N., and Cotter, A. (2011). Pegasos: Primal estimated sub-gradient solver for svm. *Mathematical programming*, 127(1):3–30.

[Shamsfard and Barforoush, 2002] Shamsfard, M. and Barforoush, A. (2002). An introduction to HASTI: an ontology learning system. In *Proceedings of the iasted international conference artificial intelligence and soft computing, Acta Press, Galgary, Canada*, pages 242–247.

[Shin et al., 2015] Shin, J., Wu, S., Wang, F., De Sa, C., Zhang, C., and Ré, C. (2015). Incremental knowledge base construction using DeepDive. *Proceedings of the VLDB Endowment*, 8(11):1310–1321.

[Shridhar et al., 2019] Shridhar, K., Laumann, F., and Liwicki, M. (2019). A comprehensive guide to bayesian convolutional neural network with variational inference. *arXiv preprint arXiv:1901.02731.*

[Sidner, 1979] Sidner, C. L. (1979). Towards a Computational Theory of Definite Anaphora Comprehension in English Discourse. Technical report, Massachusetts Inst of Tech Cambridge Artificial Intelligence lab.

[Skut et al., 1997] Skut, W., Krenn, B., Brants, T., and Uszkoreit, H. (1997). An annotation scheme for free word order languages. *arXiv preprint cmp-lg/9702004.*

[Soon et al., 2001] Soon, W. M., Ng, H. T., and Lim, D. C. Y. (2001). A machine learning approach to coreference resolution of noun phrases. *Computational linguistics*, 27(4):521–544.

[Spearman, 1904] Spearman, C. (1904). The proof and measurement of association between two things. *American journal of Psychology*, 15(1):72–101.

[Stede, 2004] Stede, M. (2004). The Potsdam commentary corpus. In *Proceedings of the 2004 ACL Workshop on Discourse Annotation*, pages 96–102. Association for Computational Linguistics.

[Stenetorp et al., 2012] Stenetorp, P., Pyysalo, S., Topić, G., Ohta, T., Ananiadou, S., and Tsujii, J. (2012). BRAT: a web-based tool for NLP-assisted text annotation. In *Proceedings of the Demonstrations at the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 102–107. Association for Computational Linguistics.

[Surdeanu et al., 2012] Surdeanu, M., Tibshirani, J., Nallapati, R., and Manning, C. D. (2012). Multi-instance multi-label learning for relation extraction. In *Proceedings of the 2012 joint conference on empirical methods in natural language processing and computational natural language learning*, pages 455–465. Association for Computational Linguistics.

[Suri and McCoy, 1994] Suri, L. Z. and McCoy, K. F. (1994). RAFT/RAPR and centering: A comparison and discussion of problems related to processing complex sentences. *Computational linguistics*, 20(2):301–317.

[Sutton and McCallum, 2012] Sutton, C. and McCallum, A. (2012). Piecewise training for undirected models. *arXiv preprint arXiv:1207.1409.*

[Sutton et al., 2012] Sutton, C., McCallum, A., et al. (2012). An introduction to conditional random fields. *Foundations and Trends® in Machine Learning*, 4(4):267–373.

[Sutton et al., 2007] Sutton, C., McCallum, A., and Rohanimanesh, K. (2007). Dynamic conditional random fields: Factorized probabilistic models for labeling and segmenting sequence data. *Journal of Machine Learning Research*, 8(Mar):693–723.

[Sutton, 2008] Sutton, C. A. (2008). Efficient training methods for conditional random fields. Technical report, MASSACHUSETTS UNIV AMHERST DEPT OF COMPUTER SCIENCE.

[Telljohann et al., 2006] Telljohann, H., Hinrichs, E. W., Kübler, S., Zinsmeister, H., and Beck, K. (2006). Stylebook for the Tübingen treebank of written German (TüBa-D/Z). In *Seminar fur Sprachwissenschaft, Universität Tübingen, Tübingen, Germany.*

[Tieleman and Hinton, 2012] Tieleman, T. and Hinton, G. (2012). Lecture 6.5—RmsProp: Divide the gradient by a running average of its recent magnitude. COURSERA: Neural Networks for Machine Learning, 2012. Technical report, TRITA-MAT-E 2017: 81 ISRN-KTH/MAT/E–17/81–SE.

[Trilcke, 2013] Trilcke, P. (2013). Social network analysis (SNA) als Methode einer textempirischen Literaturwissenschaft. *Empirie in der Literaturwissenschaft*, (8).

[Tritscher, 2016] Tritscher, J. (2016). Evaluation vonstate of the art syntaxparsern auf deutschen romanen des 16. bis 20. jahrhunderts.

[Tsochantaridis et al., 2004] Tsochantaridis, I., Hofmann, T., Joachims, T., and Altun, Y. (2004). Support vector machine learning for interdependent and structured output spaces. In *Proceedings of the twenty-first international conference on Machine learning*, page 104. ACM.

[Tu et al., 2019] Tu, N. D. T., Krug, M., and Brunner, A. (2019). Automatic recognition of direct speech without quotation marks. A rule-based approach. *Proceedings of DHD.*

[Tudorache et al., 2013] Tudorache, T., Nyulas, C., Noy, N. F., and Musen, M. A. (2013). WebProtégé: A collaborative ontology editor and knowledge acquisition tool for the web. *Semantic web*, 4(1):89–99.

[Vaswani et al., 2017] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L. u., and Polosukhin, I. (2017). Attention is all you need. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R., editors, *Advances in Neural Information Processing Systems 30*, pages 5998–6008. Curran Associates, Inc.

[Vilain et al., 1995] Vilain, M., Burger, J., Aberdeen, J., Connolly, D., and Hirschman, L. (1995). A model-theoretic coreference scoring scheme. In *Proceedings of the 6th conference on Message understanding*, pages 45–52. Association for Computational Linguistics.

[Volk and Clematide, 2001] Volk, M. and Clematide, S. (2001). Learn-Filter-Apply-Forget. Mixed Approaches to Named Entity Recognition. In *NLDB*, volume 1, pages 153–163. Citeseer.

[Walker et al., 2006] Walker, C., Strassel, S., Medero, J., and Maeda, K. (2006). ACE 2005 multilingual training corpus. *Linguistic Data Consortium, Philadelphia*, 57.

[Waumans et al., 2015] Waumans, M. C., Nicodème, T., and Bersini, H. (2015). Topology analysis of social networks extracted from literature. *PloS one*, 10(6):e0126470.

[Wehner, 2018] Wehner, N. (2018). Anwendung von neuronalen netzen bei der klassifikation von relationen in deutschen romantexten.

[Weischedel et al., 2013] Weischedel, R., Palmer, M., Marcus, M., Hovy, E., Pradhan, S., Ramshaw, L., Xue, N., Taylor, A., Kaufman, J., Franchini, M., et al. (2013). Ontonotes release 5.0 LDC2013T19. *Linguistic Data Consortium, Philadelphia, PA*.

[Wimalasuriya and Dou, 2010] Wimalasuriya, D. C. and Dou, D. (2010). Ontology-based information extraction: An introduction and a survey of current approaches.

[Wiseman et al., 2015] Wiseman, S. J., Rush, A. M., Shieber, S. M., and Weston, J. (2015). Learning anaphoricity and antecedent ranking features for coreference resolution. Association for Computational Linguistics.

[Wu and Weld, 2007] Wu, F. and Weld, D. S. (2007). Autonomously semantifying wikipedia. In *Proceedings of the sixteenth ACM conference on Conference on information and knowledge management*, pages 41–50. ACM.

[Yang et al., 2004] Yang, X., Su, J., Zhou, G., and Tan, C. L. (2004). An NP-cluster based approach to coreference resolution. In *Proceedings of the 20th international conference on Computational Linguistics*, page 226. Association for Computational Linguistics.

[Yang et al., 2019] Yang, Z., Dai, Z., Yang, Y., Carbonell, J., Salakhutdinov, R., and Le, Q. V. (2019). XLNet: Generalized Autoregressive Pretraining for Language Understanding. *arXiv preprint arXiv:1906.08237*.

[Yeung and Lee, 2017] Yeung, C. Y. and Lee, J. (2017). Identifying Speakers and Listeners of Quoted Speech in Literary Works. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, volume 2, pages 325–329.

[Yu and Kim, 2012] Yu, H. and Kim, S. (2012). SVM tutorial—classification, regression and ranking. *Handbook of Natural computing*, pages 479–506.

[Zeng et al., 2014] Zeng, D., Liu, K., Lai, S., Zhou, G., Zhao, J., et al. (2014). Relation classification via convolutional deep neural network.

[Zhang and Wang, 2015] Zhang, D. and Wang, D. (2015). Relation classification via recurrent neural network. *arXiv preprint arXiv:1508.01006*.

[Zhang et al., 2003] Zhang, J. Y., Black, A. W., and Sproat, R. (2003). Identifying speakers in children's stories for speech synthesis. In *Eighth European Conference on Speech Communication and Technology.*

[Zhang et al., 2006] Zhang, M., Zhang, J., Su, J., and Zhou, G. (2006). A composite kernel to extract relations between entities with both flat and structured features. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, pages 825–832. Association for Computational Linguistics.

[Zhao and Grishman, 2005] Zhao, S. and Grishman, R. (2005). Extracting relations with integrated information using kernel methods. In *Proceedings of the 43rd annual meeting on association for computational linguistics*, pages 419–426. Association for Computational Linguistics.

[Zhou et al., 2007] Zhou, G., Zhang, M., Ji, D., and Zhu, Q. (2007). Tree kernel-based relation extraction with context-sensitive structured parse tree information. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 728–736.

[Zhou et al., 2016] Zhou, P., Shi, W., Tian, J., Qi, Z., Li, B., Hao, H., and Xu, B. (2016). Attention-based bidirectional long short-term memory networks for relation classification. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 207–212.

[Zhou and Li, 2005] Zhou, Z.-H. and Li, M. (2005). Tri-training: Exploiting unlabeled data using three classifiers. *IEEE Transactions on Knowledge & Data Engineering*, (11):1529–1541.

[Zhu et al., 2006] Zhu, X., Kandola, J., Lafferty, J., and Ghahramani, Z. (2006). Graph kernels by spectral transforms. *Semi-supervised learning*, pages 277–291.

[Zilly et al., 2017] Zilly, J. G., Srivastava, R. K., Koutník, J., and Schmidhuber, J. (2017). Recurrent highway networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 4189–4198. JMLR. org.

[Zitnik, 2012] Zitnik, S. (2012). Collective Ontology-based Information Extraction using Probabilistic Graphical Models. In *CAiSE (Doctoral Consortium).*

[Zöllner-Weber, 2008] Zöllner-Weber, A. (2008). Noctua literaria: a computer-aided approach for the formal description of literary characters using an ontology.

A. Tagsets

A.1. STTS Tagset

Tag	Beschreibung	Beispiel
ADJA	attributives Adjektiv	[das] große [Haus]
ADJD	adverbiales oder prädikatives Adjektiv	[er fährt] schnell, [er ist] schnell
ADV	Adverb	schon, bald, doch
APPR	Präposition; Zirkumposition links	in [der Stadt], ohne [mich]
APPRART	Präposition mit Artikel	im [Haus], zur [Sache]
APPO	Postposition	[ihm] zufolge, [der Sache] wegen
APZR	Zirkumposition rechts	[von jetzt] an
ART	bestimmter oder unbestimmter Artikel	der, die, das, ein, eine
CARD	Kardinalzahl	zwei [Männer], [im Jahre] 1994
FM	Fremdsprachliches Material	A big fish
ITJ	Interjektion	mhm, ach, tja
KOUI	unterordnende Konj. mit "zu" und Infinitiv	um [zu leben], anstatt [zu fragen]
KOUS	unterordnende Konj. mit Satz	weil, dass, damit, wenn, ob
KON	nebenordnende Konj.	und, oder, aber
KOKOM	Vergleichskonjunktion	als, wie
NN	normales Nomen	Tisch, Herr, [das] Reisen
NE	Eigennamen	Hans, Hamburg, HSV
PDS	substituierendes Demonstrativpron.	dieser, jener
PDAT	attribuierendes Demonstrativpron.	jener [Mensch]
PIS	substituierendes Indefinitpron.	keiner, viele, man, niemand
PIAT	attribuierendes Indefinitpron. o. Determiner	kein [Mensch]
PIDAT	attribuierendes Indefinitpron. m. Determiner	[ein] wenig [Wasser]
PPER	irreflexives Personalpronomen	ich, er, ihm, mich, dir
PPOSS	substituierendes Possessivpron.	meins, deiner
PPOSAT	attribuierendes Possessivpron.	mein [Buch], deine [Mutter]
PRELS	substituierendes Relativpron.	[der Hund ,] der
PRELAT	attribuierendes Relativpron.	[der Mann ,] dessen [Hund]
PRF	reflexives Personalpronomen	sich, einander, dich, mir
PWS	substituierendes Interrogativpronomen	wer, was
PWAT	attribuierendes Interrogativpronomen	welche[Farbe], wessen [Hut]

Tag	Beschreibung	Beispiel
PWAV	adverbiales Interrogativ- oder Relativpron.	warum, wo,...
PAV	Pronominaladverb	dafür, dabei, deswegen, trotzdem
PTKZU	"zu" vor Infinitiv	zu [gehen]
PTKNEG	Negationspartikel	nicht
PTKVZ	abgetrennter Verbzusatz	[er kommt] an, [er fährt] rad
PTKANT	Antwortpartikel	ja, nein, danke, bitte
PTKA	Partikel bei Adjektiv oder Adverb	am [schönsten], zu [schnell]
TRUNC	Kompositions-Erstglied	An- [und Abreise]
VVFIN	finites Verb, voll	[du] gehst, [wir] kommen [an]
VVIMP	Imperativ, voll	komm [!]
VVINF	Infinitiv, voll	gehen, ankommen
VVIZU	Infinitiv mit "zu", voll	anzukommen, loszulassen
VVPP	Partizip Perfekt, voll	gegangen, angekommen
VAFIN	finites Verb, aux	[du] bist, [wir] werden
VAIMP	Imperativ, aux	sei [ruhig !]
VAINF	Infinitiv, aux	werden, sein
VAPP	Partizip Perfekt, aux	gewesen
VMFIN	finites Verb, modal	dürfen
VMINF	Infinitiv, modal	wollen
VMPP	Partizip Perfekt, modal	gekonnt, [er hat gehen] können
XY	Nichtwort, Sonderzeichen enthaltend	3:7, H2O, D2XW3
$,	Komma	,
$.	Satzbeendende Interpunktion	. ? ! ; :
$(	sonstige Satzzeichen; satzintern	- [,]()

A.2. NEGRA Dependency Edges

AC	adpositional case marker	MW	way (directional modifier)
ADC	adjective component	NG	negation
AMS	measure argument of adj	NK	noun kernel modifier
APP	apposition	NMC	numerical component
AVC	adverbial phrase component	OA	accusative object
CC	comparative complement	OA2	second accusative object
CD	coordinating conjunction	OC	clausal object
CJ	conjunct	OG	genitive object
CM	comparative concjunction	PD	predicate
CP	complementizer	PG	pseudo-genitive
DA	dative	PH	placeholder
DH	discourse-level head	PM	morphological particle
DM	discourse marker	PNC	proper noun component
GL	prenominal genitive	RC	relative clause
GR	postnominal genitive	RE	repeated element
HD	head	RS	reported speech
JU	junctor	SB	subject
MC	comitative	SBP	passivised subject (PP)
MI	instrumental	SP	subject or predicate
ML	locative	SVP	separable verb prefix
MNR	postnominal modifier	UC	(idiosyncratic) unit component
MO	modifier	VO	vocative

Table A.1.: Dependency-Relations

Table A.1 contains the dependency relations that are used by the dependency parser, a more detailed explanation of them is provided at: `http://www.coli.uni-saarland.de/projects/sfb378/negra-corpus/kanten.html` zu finden.

B. Recommended Configuration of the Kallimachos Pipeline

```
# All lines starting with a # are ignored, so uncomment if you want to skip a
    preprocessing step
#
# use this file to configure the preprocessing pipeline. This supports only
    german preprocessing so far
# If you wish to use more than 1 engine you need to separate them with a
    semicolon ';'. If not specified other, the stages overwrite each other,
    keeping only the last
#

###=================INPUT==========###
# The System can atm only support .xmi that were made with CorefTypesystem.xml
    and .txt files. It can differ between them internally. Fresh documents are
    welcome



###=================TOKENIZATION==========###
# For tokenization you can choose between:
# 1. OpenNLP_Tokenizer
# 2. RuleBased_Tokenizer
#

#TOKENIZER=OpenNLP_Tokenizer



###=================Sentence Splitting==========###
# For sentence splitting you can choose between:
# 1. OpenNLP_SentenceSplitter
# 2. RulebasedSentenceDetection
#

#SENTENCE_SPLITTER= OpenNLP_SentenceSplitter



###=================Paragraph Detection==========###
# For Paragraph Detection you can choose between:
```

```
# 1. Simple_Kernkorpus_Paragraph_Detection
# 2. Simple_Linebreak_Paragraph_Detection
#

#PARAGRAPH=Simple_Linebreak_Paragraph_Detection



###===============POS-Tagging==============###
# For POS-Tagging you can choose between
# 1. OpenNLP_POSTagger
# 2. Mate_POSTagger
# 3. TreeTagger_POSTagger

#POS_TAGGER= TreeTagger_POSTagger



###=================MORPHOLOGY==========###
# For MorphTagging you can choose between (multiselection possible):
# 1. RFT_Tagger ( Trigramm HMM Tagger of Helmut Schmid) (JNI - Requires Windows
     to the best of my knowledge)
# 2. TIGER_Morph ( This is a MaxEnt Model trained on TIGER)
# 3. Mate_Morph

#MORPH_TAGGER = RFT_Tagger;Mate_Morph



###=================Lemmatizer==========###
# For MorphTagging you can choose between (multiselection possible):
# 1. Mate_Lemmatizer
# 2. TreeTagger_Lemmatizer
#

#LEMMATIZER= TreeTagger_Lemmatizer



###=================Chunker==========###
# For Chunking (Shallow Parsing) so far only the treetagger is available
# 1. TreeTagger_Chunker
#
#
#CHUNKER= TreeTagger_Chunker



###=================Parser==========###
```

```
# For Parsing there are actually 2 included models (multiselection possible)
# Note: This step needs to have POS Tags and Lemmas
# 1. Mate_Dependency
# 2. Stanford_Constituency
# 3. Medical_Document_Constituency_Parser
# 4. Berkeley_Constituency_Parser
# 5. Parzu_Dependency (ParZu needs to be 'installed'; you can get it from https
    ://github.com/rsennrich/ParZu)
# Note: PARZU_PATH needs to be set for this to work
#

#PARSER= Parzu_Dependency

#
# Note: Only necessary if Parzu_Dependency is used as Parser
# The path to the folder where the files of ParZu are located
# the ~ character is interpreted as the user's home folder ( System.getProperty
    ("user.home") )
#
#PARZU_PATH = ~/git/ParZu/parzu


###=================Direct-Speech/Speaker-Detection=============###
#
# 1. DialogBasedSpeakerDetection
#

DIRECT_SPEECH = DialogBasedSpeakerDetection



###=================NE-Tagger==========###
# Only 1 NE-Tagger supported atm, this one requires POS-Tags, Tokens and
    Sentences
# 1. KallimachosTagger
# 2. RulebasedNE

#NE_TAGGER = RulebasedNE



###=================COREF-Resolution==============####
# For the coreference resolution this system only supports the selfmade
    Algorithm based on the stanford system
# 1. StanfordAlgorithm:
# If you do not want to apply all Sieves, list the Sieves that should be
    applied (separated by commata)
# 1) ExactStringMatchingSieve
# 2) SpeakerResolutionSieve
# 3) RelaxedStringMatchingSieve
# 4) PreciseConstructsSieve
# 5) StrictHeadMatchingSieve
# 6) StrictHeadVarBSieve
# 7) RelaxedHeadMatchingSieve
# 8) HeadWordInclusionSieve
# 9) NicknameHeadMatchingSieve
# 10) NameMergeProhibitionSieve
# 11) ChunkAttributesMatchingSieve
# 12) PronounSieve
# 13) PronounDsSieve
# 14) FamilyRelationSieve
# 15) PossessedAppellativeSieve
# 2. NamelistAlgorithm

COREF_RESOLVER = StanfordAlgorithm,1,2,3,4,5,6,9,10,11,12,13,14,15



###================Relation-Detection=================###
#
# 1. MaxEntRelationDetection
# 2. HierMaxEntInteractioNDetection
#

#RELATION_DETECTION= MaxEntRelationDetection



###=================Information-Extraction==============####
#
# 1. AttributeExtraction
#

#INFORMATION_EXTRACTION = AttributeExtraction



###=================Document-Partitioning==============####
#
# 1. DocumentPartitioner: can specify type that is used for
# partitioning by appending ',<typename>', e.g. ',de.uniwue.kalimachos.coref.
    type.Sentence'
# default type: "de.uniwue.kalimachos.coref.type.Paragraph"
#
# Note that you should probably use ENGINE_ORDER to ensure the partitioner
# is used at the right point in the pipeline
#
```

```
#DOCUMENT_PARTITIONING = DocumentPartitioner



###================Engine-Order=================###
#
# Usually the order of the used engines is computed automatically.
# You can use this property to enforce the engine order you want.
# (case where this is necessary: splitting of documents)
# NOTE: engines listed here are used in the pipeline even if they are
# not listed in the section they belong to
#

#ENGINE_ORDER =



##=======Output=========###
# Choose one of the following formats as output
#
# 1. XMI_Coref
# 2. JTF (Kallimachos Tab format)
# 3. TCF (Weblicht XML)
# 4. CONLL ( currently unsupported)

OUTPUT_FORMAT= XMI_Coref



###========File-Skipping==========###
#
# Choose whether to skip files that already exist in the target
# folder or overwrite them (skip = true, overwrite = false)
#
# default behaviour: skip existing files
#

#SKIP_EXISTING_FILES = false
```

C. Apache UIMA typesystem with most relevant types for this work

The current section describes the Apache UIMA typesystem which is used throughout this project.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<typeSystemDescription xmlns="http://uima.apache.org/resourceSpecifier">
 <name>Kallimachos</name>
 <description />
 <version>1.0</version>
 <vendor />
 <types>
  <typeDescription>
   <name>de.uniwue.kalimachos.coref.type.NamedEntity</name>
   <description />
   <supertypeName>uima.tcas.Annotation</supertypeName>
   <features>
    <featureDescription>
     <name>Name</name>
     <description />
     <rangeTypeName>uima.cas.String</rangeTypeName>
    </featureDescription>
    <featureDescription>
     <name>ID</name>
     <description />
     <rangeTypeName>uima.cas.String</rangeTypeName>
    </featureDescription>
    <featureDescription>
     <name>SystemID</name>
     <description />
     <rangeTypeName>uima.cas.String</rangeTypeName>
    </featureDescription>
    <featureDescription>
     <name>MappedID</name>
     <description />
     <rangeTypeName>uima.cas.String</rangeTypeName>
    </featureDescription>
    <featureDescription>
 <name>CoreNamedEntity</name>
 <description>The Type of the annotation, vaid values are:
  Core//Appleativ Abstrakta//Appelativ: part of fictional world
 </description>
 <rangeTypeName>uima.cas.String</rangeTypeName>
</featureDescription>
<featureDescription>
 <name>Numerus</name>
 <description />
 <rangeTypeName>uima.cas.String</rangeTypeName>
</featureDescription>
<featureDescription>
 <name>Pseudo</name>
 <description />
 <rangeTypeName>uima.cas.String</rangeTypeName>
</featureDescription>
<featureDescription>
 <name>CoreRange</name>
 <description>Contains the Span that represents the Core-Feature
 </description>
 <rangeTypeName>uima.cas.String</rangeTypeName>
</featureDescription>
<featureDescription>
 <name>Gender</name>
 <description />
 <rangeTypeName>uima.cas.String</rangeTypeName>
</featureDescription>
<featureDescription>
 <name>Uncertain</name>
 <description />
 <rangeTypeName>uima.cas.String</rangeTypeName>
</featureDescription>
<featureDescription>
 <name>NEType</name>
 <description>Appelativ Abstrakta // Core // Appleativ teil der
  fikt. Welt
 </description>
 <rangeTypeName>uima.cas.String</rangeTypeName>
</featureDescription>
<featureDescription>
 <name>IsImportant</name>
 <description />
 <rangeTypeName>uima.cas.String</rangeTypeName>
</featureDescription>
```

```
  <featureDescription>
   <name>ResolvedReference</name>
   <description />
   <rangeTypeName>uima.tcas.Annotation</rangeTypeName>
  </featureDescription>
  <featureDescription>
   <name>Person</name>
   <description />
   <rangeTypeName>uima.cas.String</rangeTypeName>
  </featureDescription>
  <featureDescription>
   <name>Construction</name>
   <description />
   <rangeTypeName>uima.cas.String</rangeTypeName>
  </featureDescription>
 </features>
</typeDescription>
<typeDescription>
 <name>de.uniwue.kalimachos.coref.type.Sentence</name>
 <description />
 <supertypeName>uima.tcas.Annotation</supertypeName>
 <features>
  <featureDescription>
   <name>ConstituencyParse</name>

   <rangeTypeName>uima.cas.String</rangeTypeName>
  </featureDescription>
  <featureDescription>
   <name>Dialekt</name>
   <description />
   <rangeTypeName>uima.cas.String</rangeTypeName>
  </featureDescription>
 </features>
</typeDescription>
<typeDescription>
 <name>de.uniwue.kalimachos.coref.type.Token</name>
 <description />
 <supertypeName>uima.tcas.Annotation</supertypeName>
</typeDescription>
<typeDescription>
 <name>de.uniwue.kalimachos.coref.type.POS</name>
 <description />
 <supertypeName>uima.tcas.Annotation</supertypeName>
 <features>
```

```
<featureDescription>
 <name>POSTag</name>
 <description />
 <rangeTypeName>uima.cas.String</rangeTypeName>
</featureDescription>
<featureDescription>
 <name>Lemma</name>
 <description />
 <rangeTypeName>uima.cas.String</rangeTypeName>
</featureDescription>
<featureDescription>
 <name>Stem</name>
 <description />
 <rangeTypeName>uima.cas.String</rangeTypeName>
</featureDescription>
<featureDescription>
 <name>CompoundSplit</name>
 <description>Has the split of a compound word in the form
  &lt;part&gt; &lt;part&gt;...</description>
 <rangeTypeName>uima.cas.String</rangeTypeName>
</featureDescription>
<featureDescription>
 <name>DialektArray</name>
 <description />
 <rangeTypeName>uima.cas.StringArray</rangeTypeName>
</featureDescription>
<featureDescription>
 <name>CoveringSentence</name>
 <description />
 <rangeTypeName>uima.tcas.Annotation</rangeTypeName>
</featureDescription>
<featureDescription>
 <name>ClassifierTemp</name>
 <description>This field can be abused to store information in the
  token
 </description>
 <rangeTypeName>uima.cas.String</rangeTypeName>
</featureDescription>
<featureDescription>
 <name>Confidence</name>
 <description />
 <rangeTypeName>uima.cas.String</rangeTypeName>
</featureDescription>
<featureDescription>
```

```
   <name>ClassifierTemp2</name>
   <description />
   <rangeTypeName>uima.cas.String</rangeTypeName>
  </featureDescription>
  <featureDescription>
   <name>ClassifierTemp3</name>
   <description />
   <rangeTypeName>uima.cas.String</rangeTypeName>
  </featureDescription>
 </features>
</typeDescription>
<typeDescription>
 <name>de.uniwue.kalimachos.coref.type.Chunk</name>
 <description />
 <supertypeName>uima.tcas.Annotation</supertypeName>
 <features>
  <featureDescription>
   <name>ChunkType</name>
   <description />
   <rangeTypeName>uima.cas.String</rangeTypeName>
  </featureDescription>
 </features>
</typeDescription>
<typeDescription>
 <name>de.uniwue.kalimachos.coref.type.DependencyParse</name>
 <description />
 <supertypeName>uima.tcas.Annotation</supertypeName>
 <features>
  <featureDescription>
   <name>Headname</name>
   <description />
   <rangeTypeName>uima.cas.String</rangeTypeName>
  </featureDescription>
  <featureDescription>
   <name>WordNumber</name>
   <description />
   <rangeTypeName>uima.cas.Integer</rangeTypeName>
  </featureDescription>
  <featureDescription>
   <name>DependencyRelation</name>
   <description />
   <rangeTypeName>uima.cas.String</rangeTypeName>
  </featureDescription>
  <featureDescription>
```

```
     <name>HeadAnnotation</name>

     <rangeTypeName>uima.tcas.Annotation</rangeTypeName>
    </featureDescription>
   </features>
  </typeDescription>
  <typeDescription>
   <name>de.uniwue.kalimachos.coref.type.StanfordParse</name>
   <description />
   <supertypeName>uima.tcas.Annotation</supertypeName>
   <features>
    <featureDescription>
     <name>Children</name>
     <description />
     <rangeTypeName>uima.cas.String</rangeTypeName>
    </featureDescription>
    <featureDescription>
     <name>PhraseType</name>
     <description />
     <rangeTypeName>uima.cas.String</rangeTypeName>
    </featureDescription>
    <featureDescription>
     <name>ChildrenAnnos</name>
     <description />
     <rangeTypeName>uima.cas.FSArray</rangeTypeName>
     <elementType>uima.tcas.Annotation</elementType>
    </featureDescription>
    <featureDescription>
     <name>Parent</name>
     <description />
     <rangeTypeName>uima.tcas.Annotation</rangeTypeName>
    </featureDescription>
   </features>
  </typeDescription>
  <typeDescription>
   <name>de.uniwue.kalimachos.coref.type.PolarityClue</name>
   <description />
   <supertypeName>uima.tcas.Annotation</supertypeName>
   <features>
    <featureDescription>
     <name>polarityClue</name>
     <description />
     <rangeTypeName>uima.cas.String</rangeTypeName>
    </featureDescription>
```

```
 </features>
</typeDescription>
<typeDescription>
 <name>de.uniwue.kalimachos.coref.type.DirectSpeech</name>
 <description />
 <supertypeName>uima.tcas.Annotation</supertypeName>
 <features>
  <featureDescription>
   <name>ID</name>
   <description />
   <rangeTypeName>uima.cas.String</rangeTypeName>
  </featureDescription>
  <featureDescription>
   <name>Speaker</name>
   <description />
   <rangeTypeName>uima.tcas.Annotation</rangeTypeName>
  </featureDescription>
  <featureDescription>
   <name>SpokenTo</name>
   <description />
   <rangeTypeName>uima.tcas.Annotation</rangeTypeName>
  </featureDescription>
  <featureDescription>
   <name>SpeakerArray</name>
   <description />
   <rangeTypeName>uima.cas.FSArray</rangeTypeName>
   <elementType>uima.tcas.Annotation</elementType>
  </featureDescription>
  <featureDescription>
   <name>SpokenToArray</name>
   <description />
   <rangeTypeName>uima.cas.FSArray</rangeTypeName>
   <elementType>uima.tcas.Annotation</elementType>
  </featureDescription>
  <featureDescription>
   <name>Category</name>
   <description />
   <rangeTypeName>uima.cas.String</rangeTypeName>
  </featureDescription>
  <featureDescription>
   <name>UnknownSpeaker</name>
   <description />
   <rangeTypeName>uima.cas.String</rangeTypeName>
  </featureDescription>
```

```
    <featureDescription>
     <name>UnknownSpokenTo</name>
     <description />
     <rangeTypeName>uima.cas.String</rangeTypeName>
    </featureDescription>
   </features>
  </typeDescription>
  <typeDescription>
   <name>de.uniwue.kalimachos.coref.type.Chapter</name>
   <description />
   <supertypeName>uima.tcas.Annotation</supertypeName>
   <features>
    <featureDescription>
     <name>Enumeration</name>
     <description />
     <rangeTypeName>uima.cas.String</rangeTypeName>
    </featureDescription>
   </features>
  </typeDescription>
  <typeDescription>
   <name>de.uniwue.kalimachos.coref.type.Paragraph</name>
   <description />
   <supertypeName>uima.tcas.Annotation</supertypeName>
   <features>
    <featureDescription>
     <name>PragraphType</name>
     <description />
     <rangeTypeName>uima.cas.String</rangeTypeName>
    </featureDescription>
       <featureDescription>
     <name>MetaInfo</name>
     <description />
     <rangeTypeName>uima.cas.String</rangeTypeName>
    </featureDescription>
   </features>
  </typeDescription>
  <typeDescription>
   <name>de.uniwue.kalimachos.coref.type.Interaktion</name>
   <description />
   <supertypeName>uima.tcas.Annotation</supertypeName>
   <features>
    <featureDescription>
     <name>Agent</name>
     <description />
```

```
      <rangeTypeName>uima.tcas.Annotation</rangeTypeName>
     </featureDescription>
     <featureDescription>
      <name>Recipient</name>
      <description />
      <rangeTypeName>uima.tcas.Annotation</rangeTypeName>
     </featureDescription>
    </features>
   </typeDescription>
   <typeDescription>
    <name>de.uniwue.kalimachos.coref.type.RFTagType</name>
    <description />
    <supertypeName>uima.tcas.Annotation</supertypeName>
    <features>
     <featureDescription>
      <name>Tag</name>
      <description />
      <rangeTypeName>uima.cas.String</rangeTypeName>
     </featureDescription>
    </features>
   </typeDescription>
   <typeDescription>
    <name>de.uniwue.kalimachos.coref.type.Relation</name>
    <description />
    <supertypeName>uima.tcas.Annotation</supertypeName>
    <features>
     <featureDescription>
      <name>Agens</name>
      <description />
      <rangeTypeName>uima.tcas.Annotation</rangeTypeName>
     </featureDescription>
     <featureDescription>
      <name>Agens2</name>
      <description />
      <rangeTypeName>uima.tcas.Annotation</rangeTypeName>
     </featureDescription>
     <featureDescription>
      <name>Type</name>
      <description />
      <rangeTypeName>uima.cas.String</rangeTypeName>
     </featureDescription>
     <featureDescription>
      <name>Description</name>
      <description />
    <rangeTypeName>uima.cas.String</rangeTypeName>
   </featureDescription>
   <featureDescription>
    <name>Confidence</name>
    <description />
    <rangeTypeName>uima.cas.String</rangeTypeName>
   </featureDescription>
   <featureDescription>
    <name>Attribut</name>
    <description />
    <rangeTypeName>uima.tcas.Annotation</rangeTypeName>
   </featureDescription>
   <featureDescription>
    <name>Subjectivity</name>

    <rangeTypeName>uima.cas.String</rangeTypeName>
   </featureDescription>
   <featureDescription>
    <name>Tempus</name>
    <description>Ist das Attribut aktuell vergangen oder in der zukunft
    </description>
    <rangeTypeName>uima.cas.String</rangeTypeName>
   </featureDescription>
   <featureDescription>
    <name>Explicity</name>
    <description>ist das Attribut explizit genannt oder implizit
    </description>
    <rangeTypeName>uima.cas.String</rangeTypeName>
   </featureDescription>
   <featureDescription>
    <name>Truth</name>
    <description>Kann man damit rechnen, dass das Attribut der Wahrheit
     entspricht ?
    </description>
    <rangeTypeName>uima.cas.String</rangeTypeName>
   </featureDescription>
   <featureDescription>
    <name>Konjunktiv</name>
    <rangeTypeName>uima.cas.String</rangeTypeName>
   </featureDescription>
  </features>
 </typeDescription>
 <typeDescription>
  <name>de.uniwue.kallimachos.coref.type.Scene</name>
 <description />
 <supertypeName>uima.tcas.Annotation</supertypeName>
 <features>
  <featureDescription>
   <name>SceneMarker</name>
   <description>All Scenemarker in this scene, they indicate the
    beginning or end of this scene
   </description>
   <rangeTypeName>uima.cas.FSArray</rangeTypeName>
   <elementType>de.uniwue.kallimachos.coref.type.SceneMarker
   </elementType>
  </featureDescription>
  <featureDescription>
   <name>Type</name>
   <description />
   <rangeTypeName>uima.cas.String</rangeTypeName>
  </featureDescription>
  <featureDescription>
   <name>Link</name>

   <rangeTypeName>de.uniwue.kalimachos.coref.type.Scene
   </rangeTypeName>
  </featureDescription>
  <featureDescription>
   <name>Name</name>
   <description />
   <rangeTypeName>uima.cas.String</rangeTypeName>
  </featureDescription>
  <featureDescription>
   <name>Covered</name>

   <rangeTypeName>uima.cas.String</rangeTypeName>
  </featureDescription>
 </features>
</typeDescription>
<typeDescription>
 <name>de.uniwue.kallimachos.coref.type.SceneMarker</name>
 <description />
 <supertypeName>uima.tcas.Annotation</supertypeName>
 <features>
  <featureDescription>
   <name>Type</name>
   <description />
   <rangeTypeName>uima.cas.String</rangeTypeName>
```

```
    </featureDescription>
   </features>
  </typeDescription>
  <typeDescription>
   <name>de.uniwue.mk.kallimachos.ie.IEEntity</name>
   <description />
   <supertypeName>uima.tcas.Annotation</supertypeName>
   <features>
    <featureDescription>
     <name>Category</name>
     <description />
     <rangeTypeName>uima.cas.String</rangeTypeName>
    </featureDescription>
    <featureDescription>
     <name>ConstructedBy</name>
     <description />
     <rangeTypeName>uima.cas.String</rangeTypeName>
    </featureDescription>
    <featureDescription>
     <name>datatype</name>
     <description />
     <rangeTypeName>uima.cas.String</rangeTypeName>
    </featureDescription>
    <featureDescription>
     <name>owlid</name>
     <description />
     <rangeTypeName>uima.cas.String</rangeTypeName>
    </featureDescription>
    <featureDescription>
     <name>postprocessing</name>
     <description />
     <rangeTypeName>uima.tcas.Annotation</rangeTypeName>
    </featureDescription>
   </features>
  </typeDescription>
  <typeDescription>
   <name>de.uniwue.mk.kallimachos.ie.IERelation</name>
   <description />
   <supertypeName>uima.tcas.Annotation</supertypeName>
   <features>
    <featureDescription>
     <name>EntityFrom</name>
     <description />
     <rangeTypeName>uima.tcas.Annotation</rangeTypeName>
    </featureDescription>
    <featureDescription>
     <name>EntityTo</name>
     <description />
     <rangeTypeName>uima.tcas.Annotation</rangeTypeName>
    </featureDescription>
    <featureDescription>
     <name>Label</name>
     <description />
     <rangeTypeName>uima.cas.String</rangeTypeName>
    </featureDescription>
    <featureDescription>
     <name>DisplayLabel</name>
     <description />
     <rangeTypeName>uima.cas.String</rangeTypeName>
    </featureDescription>
   </features>
  </typeDescription>
  <typeDescription>
   <name>de.uniwue.mk.kallimachos.ie.IERelationGold</name>
   <description />
   <supertypeName>uima.tcas.Annotation</supertypeName>
   <features>
    <featureDescription>
     <name>EntityFrom</name>
     <description />
     <rangeTypeName>uima.tcas.Annotation</rangeTypeName>
    </featureDescription>
    <featureDescription>
     <name>EntityTo</name>
     <description />
     <rangeTypeName>uima.tcas.Annotation</rangeTypeName>
    </featureDescription>
    <featureDescription>
     <name>Label</name>
     <description />
     <rangeTypeName>uima.cas.String</rangeTypeName>
    </featureDescription>
    <featureDescription>
     <name>DisplayLabel</name>
     <description />
     <rangeTypeName>uima.cas.String</rangeTypeName>
    </featureDescription>
   </features>
  </typeDescription>
  <typeDescription>
   <name>de.uniwue.mk.kall.Dialog</name>
   <description />
   <supertypeName>uima.tcas.Annotation</supertypeName>
   <features>
    <featureDescription>
     <name>Sprechakte</name>

     <rangeTypeName>uima.cas.FSArray</rangeTypeName>
     <elementType>uima.tcas.Annotation</elementType>
    </featureDescription>
   </features>
  </typeDescription>
 </types>
</typeSystemDescription>
```

D. Rulesets for different algorithms that were created throughout this work

D.1. Ruleset for the Detection of Family, Love and Social Relations

```
    <NE> angehörte <NE> hatUntergebenen(2,1)
Oberin hatte mit <NE> hatUntergebenen(1,2)
Regiment hatUntergebenen(1,2)
Volke hatUntergebenen(1,2)

Tante hatTante(1,2)
ich bin , Tante hatTante(1,2)
Tante <GENITIV> hatTante(2,1)

<BERUF> hatDienstleister(1,2)
Anwalts hatDienstleister(1,2)
Pflegerin hatDienstleister(1,2)
Anwalt hatDienstleister(1,2)
<GENITIV> <BERUF> hatDienstleister(1,2)
<NE> pflegt <NE> hatDienstleister(1,2)
Schreiber hatDienstleister(1,2)
Knechten hatDienstleister(1,2)

<NE> Zwillinge und <NE> hatGeschwisterteil(1,2)
Sohn und eine Tochter hatGeschwisterteil(1,2)

<NE> Adoption durch <NE> hatPflegeelternteil(1,2)

<NE> eine Tochter hatTochter(1,2)
<NE> gebiert Tochter hatTochter(1,2)
Tochter eines <GENITIV> hatTochter(1,2)
beiden Töchtern hatTochter(1,2)
<ENDS-S> Tochter hatTochter(1,2)
Mutter ermahnte sie hatTochter(1,2)
Mutter und Tochter hatTochter(1,2)
<NE> geschenkt Tochter hatTochter(1,2)
<NE> die Tochter hatTochter(1,2)
Tochter <GENITIV> hatTochter(1,2)
dessen Tochter hatTochter(1,2)
Tochter hatTochter(1,2)
Tochter lässt <NE> hatTochter(1,2)
Töchter hatTochter(1,2)
<GENITIV> Tochter hatTochter(1,2)

Knaben hatSohn(1,2)
Eltern solchen Sohnes hatSohn(1,2)
<NE> haben Sohn hatSohn(1,2)
Sohn <GENITIV> hatSohn(2,1)
Buben hatSohn(1,2)
Sohn eines <NE> hatSohn(2,1)
Sohne hatSohn(1,2)
Vater seines Freundes hatSohn(1,2)
<NE> gebiert Sohn hatSohn(1,2)
<NE> beim Sohn hatSohn(1,2)
Söhne des <NE> hatSohn(1,2)
Sohn des <NE> hatSohn(1,2)
Sohn hatSohn(1,2)
Vater weist ihn hatSohn(1,2)
dessen Sohn hatSohn(1,2)
Söhnen hatSohn(1,2)
<GENITIV> Knabe hatSohn(1,2)

Magd der <NE> hatGebieter(1,2)
<NE> des <ADELSTITEL> hatGebieter(1,2)
Gebieterinn hatGebieter(1,2)
Fürsten hatGebieter(1,2)
<GENITIV> Herr hatGebieter(1,2)
Gebieterin hatGebieter(1,2)
<GENITIV> Eigentümerin hatGebieter(1,2)
König hatGebieter(1,2)
Herrn hatGebieter(1,2)
Volker hatGebieter(2,1)
Gebieters hatGebieter(1,2)
<ADELSTITEL> hatGebieter(1,2)
Sein Herr hatGebieter(1,2)
Magd des <NE> hatGebieter(1,2)
Feldherr <GENITIV> hatGebieter(1,2)
Schutzbefohlene hatGebieter(1,2)
sein Herr hatGebieter(1,2)
unsern Herrn hatGebieter(1,2)
Knecht bei <NE> hatGebieter(1,2)
Herrin hatGebieter(1,2)
<NE> an Oberinnen hatGebieter(1,2)
<NE> mit Oberin hatGebieter(1,2)
<NE> mit , Oberin hatGebieter(1,2)
Volk hatGebieter(1,2)

Widersacher hatRivale(1,2)
```

```
<NE> bittet den Bruder hatBruder(1,2)
Halbbruders hatBruder(1,2)
Bruder hatBruder(1,2)
<GENITIV> Bruder hatBruder(1,2)
Bruder des <NE> hatBruder(1,2)
Bruder <GENITIV> hatBruder(1,2)
Bruders hatBruder(1,2)

Vorfahren hatVorfahre(1,2)

<NE> ein Schwiegersohn hatSchwiegerkind(1,2)
Schwiegersohn haben den alten <NE> hatSchwiegerkind(1,2)

Lehrherrn hatLehrer(1,2)
<ENDS-S> Mentor hatLehrer(1,2)

Ziehbruder hatStiefgeschwisterteil(1,2)

Verlobten hatVerlobten(1,2)
<ENDS-S> Verlobter hatVerlobten(1,2)
<GENITIV> Verlobter hatVerlobten(1,2)
Zukünftiger hatVerlobten(1,2)
<PRONOUN> Verlobter hatVerlobten(1,2)

Onkels hatOnkel(1,2)
Nichte des <NE> hatOnkel(1,2)
Oheims hatOnkel(1,2)

Großvater hatGrosselternteil(1,2)
<NE> den Großvater hatGrosselternteil(1,2)
<NE> bei Großmama hatGrosselternteil(1,2)

<GENITIV> Nichte hatNichte(1,2)

deren Ehe mit <NE> hatEhemann(1,2)
<NE> und Ehemann hatEhemann(1,2)
<NE> Heirat mit <NE> hatEhemann(1,2)
Mannes hatEhemann(1,2)
, Mann hatEhemann(1,2)
Bräutigam hatEhemann(1,2)
<NE> Ehe mit <NE> hatEhemann(1,2)
Gatten hatEhemann(1,2)
Witwe eines <NE> hatEhemann(1,2)
Mann hatEhemann(1,2)
Frau <GENITIV> hatEhemann(1,2)
deren Mann hatEhemann(1,2)
<NE> heiratet er hatEhemann(1,2)
Manne hatEhemann(1,2)
Frauen solcher Männer hatEhemann(1,2)
```

```
Zukünftige hatVerlobte(1,2)

<GENITIV> Neffe hatNeffen(1,2)
dessen Neffen hatNeffen(1,2)
Bruderskind hatNeffen(1,2)

<GENITIV> Dorfsekretär hatAngestellten(1,2)
Mägde hatAngestellten(1,2)
<NE> schickte <APPELLATIV> hatAngestellten(1,2)
Ausgeberin hatAngestellten(1,2)
meine <BERUF> hatAngestellten(1,2)
Leuten hatAngestellten(1,2)
Arbeiter hatAngestellten(1,2)
<NE> auf die Haushälterin hatAngestellten(1,2)
Knappen hatAngestellten(1,2)
Magd hatAngestellten(1,2)
<GENITIV> Köchin hatAngestellten(1,2)
dessen Ratgeber hatAngestellten(1,2)
Gehilfe hatAngestellten(1,2)
<NE> gewinnen <BERUF> hatAngestellten(1,2)
<NE> Vermittlung Magd hatAngestellten(1,2)
Untergebenen hatAngestellten(1,2)
Angestellten hatAngestellten(1,2)
Herr und <BERUF> hatAngestellten(1,2)
<GENITIV> Sekretär hatAngestellten(1,2)

Schülerin hatSchueler(1,2)
seinem Schützling hatSchueler(1,2)
Lehrers , den Schülern hatSchueler(1,2)
Schulmeister zu <NE> hatSchueler(1,2)
Schüler hatSchueler(1,2)

Gemahlin hatEhefrau(1,2)
Mann und Frau hatEhefrau(1,2)
er heiratet <NE> hatEhefrau(1,2)
Braut hatEhefrau(1,2)
<NE> hat geheiratet <NE> hatEhefrau(1,2)
<GENITIV> Weib hatEhefrau(1,2)
Weib hatEhefrau(1,2)
Vater widerspricht , Mutter hatEhefrau(1,2)
<NE> von Geburt bis zur Heirat mit der <NE> hatEhefrau(1,2)
, Frau hatEhefrau(1,2)
<GENITIV> Frau hatEhefrau(1,2)
dessen Braut hatEhefrau(1,2)
<NE> , , als Gemahlin hatEhefrau(1,2)
<GENITIV> Braut hatEhefrau(1,2)
Gattin hatEhefrau(1,2)
Frau hatEhefrau(1,2)
```

```
Ehefrau hatEhefrau(1,2)
Männer seiner Töchter hatEhefrau(1,2)
<NE> Sehnsucht nach Frau hatEhefrau(1,2)

<NE> diesen Feind hatFeind(1,2)
<NE> wollten Feind hatFeind(1,2)
<NE> hofft besiegen <NE> hatFeind(1,2)
<NE> hassten <NE> hatFeind(1,2)
<NE> haben Feinde hatFeind(1,2)
Feind hatFeind(1,2)
Erbfeind der <NE> hatFeind(2,1)
Nebenbuhler hatFeind(1,2)
<NE> schafft Feind hatFeind(1,2)
Gegenspieler hatFeind(1,2)
<NE> stellt sich <NE> hatFeind(1,2)
<NE> angreift <NE> hatFeind(1,2)
<ENDS-S> Antipodin hatFeind(1,2)

<NE> sich mit Kind hatKind(1,2)
Kinde hatKind(1,2)
Kind der <NE> hatKind(2,1)
<NE> hat Kind hatKind(1,2)
<NE> konnte nähren Kind hatKind(1,2)
Ich hab ein Kind hatKind(1,2)
seiner Frau und Kindern hatKind(1,2)
<NE> wünschte Kind hatKind(1,2)
Kindern des <NE> hatKind(2,1)
Kinder hatKind(1,2)
Kindes hatKind(1,2)
Kindern hatKind(1,2)
Kind hatKind(1,2)
<NE> und Kindern hatKind(1,2)
<NE> ein Kind hatKind(1,2)
Kind von <NE> hatKind(1,2)
<NE> mit Kindern hatKind(1,2)
<NE> Sorge um Kind hatKind(1,2)
<NE> hatte geboren Kind hatKind(1,2)
Kind einer <NE> hatKind(2,1)
<NE> werde geboren Kind hatKind(1,2)
<NE> Sehnsucht nach und Kindern hatKind(1,2)
<NE> , Kinder hatKind(1,2)
Kind eines <NE> hatKind(2,1)
, , Kind hatKind(1,2)

Tochter der , <NE> hatTochter(2,1)

Nachkommen hatNachfahre(1,2)

Pflegemutter hatPflegemutter(1,2)
```

```
<NE> mit Eltern hatElternteil(1,2)
Eltern hatElternteil(1,2)
<GENITIV> Eltern hatElternteil(1,2)
Eltern , und <NE> hatElternteil(1,2)
Eltern , <NE> hatElternteil(1,2)

<NE> entstammt Familie hatFamilienRelation(1,2)
<GENITIV> Familie hatFamilienRelation(1,2)
<NE> erlebt Verfall in der Familie hatFamilienRelation(1,2)
Familie eines <NE> hatFamilienRelation(2,1)
Familie des <NE> hatFamilienRelation(1,2)
<NE> redet in Familie hatFamilienRelation(1,2)
Familie hatFamilienRelation(1,2)

Diener hatDiener(1,2)
Pflegerin hatDiener(1,2)
Knecht hatDiener(1,2)
<NE> sich und Dienerin hatDiener(1,2)
<NE> eine Dirn hatDiener(1,2)
Diener eines <NE> hatDiener(2,1)
<NE> mit Gehilfen hatDiener(1,2)
beide , mit Diener hatDiener(1,2)
Herr und Diener hatDiener(1,2)

Schwester <GENITIV> hatSchwester(1,2)
Schwester seines <NE> hatSchwester(2,1)
<NE> Töchter und <NE> hatSchwester(1,2)
<NE> Schwestern und <NE> hatSchwester(1,2)
dessen Schwestern hatSchwester(1,2)
Schwester hatSchwester(1,2)
<GENITIV> Schwester hatSchwester(1,2)

Meister hatMeister(1,2)

<NE> den Adoptivsohn hatPflegekind(1,2)
Pflegetochter hatPflegekind(1,2)

Vaters hatVater(1,2)
Sohn des <NE> hatVater(1,2)
Vater des <NE> hatVater(1,2)
er gegen den Vater hatVater(1,2)
<NE> begleitet und Vater hatVater(1,2)
Sohn dem Alten hatVater(1,2)
<GENITIV> Vater hatVater(1,2)
Vater hatVater(1,2)
Vater seines <NE> hatVater(2,1)
<NE> kehrt verrichten beim Vater hatVater(1,2)
<NE> will Vater hatVater(1,2)
```

```
Kameraden hatArbeitskollegen(1,2)
Komplizin <ENDS-S> hatArbeitskollegen(1,2)
Kollegen hatArbeitskollegen(1,2)
Mitarbeiterinnen hatArbeitskollegen(1,2)

<NE> waren war <FAMILY_RELATION> hatFamilienRelation(1,2)
<FAMILY_RELATION> von diesem <NE> hatFamilienRelation(2,1)
<NE> , wird von <FAMILY_RELATION> hatFamilienRelation(1,2)
<FAMILY_RELATION> hatFamilienRelation(1,2)
<FAMILY_RELATION> , sagte <NE> hatFamilienRelation(2,1)
<FAMILY_RELATION> von <NE> hatFamilienRelation(2,1)
<NE> konnte <FAMILY_RELATION> hatFamilienRelation(1,2)
er fiel <FAMILY_RELATION> hatFamilienRelation(1,2)
<NE> könnte <FAMILY_RELATION> hatFamilienRelation(1,2)
<FAMILY_RELATION> jenes <NE> hatFamilienRelation(2,1)
<FAMILY_RELATION> und <FAMILY_RELATION> hatFamilienRelation(1,2)
<FAMILY_RELATION> des <NE> hatFamilienRelation(1,2)
<NE> an <FAMILY_RELATION> hatFamilienRelation(1,2)
<NE> suchte wegen <FAMILY_RELATION> hatFamilienRelation(1,2)
seine Sehnsucht und <FAMILY_RELATION> hatFamilienRelation(1,2)
<NE> eine treue <FAMILY_RELATION> hatFamilienRelation(1,2)
<NE> dem <FAMILY_RELATION> hatFamilienRelation(1,2)
<ENDS-S> <FAMILY_RELATION> hatFamilienRelation(1,2)
<FAMILY_RELATION> der <NE> hatFamilienRelation(1,2)
<NE> gegen <FAMILY_RELATION> hatFamilienRelation(1,2)
<NE> , wenn <FAMILY_RELATION> hatFamilienRelation(1,2)
<NE> zu <FAMILY_RELATION> hatFamilienRelation(1,2)
<FAMILY_RELATION> <GENITIV> hatFamilienRelation(1,2)
<NE> der <FAMILY_RELATION> hatFamilienRelation(1,2)
<NE> beide <FAMILY_RELATION> hatFamilienRelation(1,2)
<GENITIV> <FAMILY_RELATION> hatFamilienRelation(1,2)
<FAMILY_RELATION> dieser <NE> hatFamilienRelation(2,1)
<FAMILY_RELATION> eines <GENITIV> hatFamilienRelation(1,2)
<FAMILY_RELATION> hatFamilienRelation(1,2)
<FAMILY_RELATION> <GENITIV_NN> hatFamilienRelation(1,2)
<GENITIV_NN> <FAMILY_RELATION> hatFamilienRelation(1,2)

<GENITIV> Mutter hatMutter(1,2)
Sohn eines und Mutter hatMutter(1,2)
Mutter <ENDS-S> hatMutter(2,1)
Sohn war ihrer hatMutter(1,2)
<NE> waren Mutter hatMutter(1,2)
<NE> die Mutter hatMutter(1,2)
<NE> ist Mutter hatMutter(1,2)
<NE> Kindheit endet Mutter hatMutter(1,2)
Mutter hatMutter(1,2)
<NE> saßen Mutter hatMutter(1,2)
<NE> , wird von Mutter hatMutter(1,2)
```

```
Ziehvaters hatPflegeelternteil(1,2)

Führer hatVorgesetzten(1,2)
<BERUF> des Herrn hatVorgesetzten(1,2)
Vorgesetzten hatVorgesetzten(1,2)
Vorsteher hatVorgesetzten(1,2)
Hauptmann hatVorgesetzten(1,2)
Magd deines <NE> hatVorgesetzten(1,2)
Herrin hatVorgesetzten(1,2)
<NE> tritt in Dienste <NE> hatVorgesetzten(1,2)
Brotherrn hatVorgesetzten(1,2)
<NE> läuft und tritt in Dienst <NE> hatVorgesetzten(1,2)
<NE> in Dienst des <NE> hatVorgesetzten(1,2)
<BERUF> von <NE> hatVorgesetzten(1,2)
Novizen hatVorgesetzten(1,2)
<NE> und Advokaten hatVorgesetzten(1,2)
Oberin hatVorgesetzten(1,2)
Magd des <NE> hatVorgesetzten(1,2)
<NE> der Kaiser hatVorgesetzten(1,2)
<NE> hatte vorgestellt <ADELSTITEL> hatVorgesetzten(1,2)

Mutter und Tante hatVerwandten(1,2)
Base hatVerwandten(1,2)
Verwandten meines <NE> hatVerwandten(1,2)
<NE> einen Vetter hatVerwandten(1,2)
Angehörigen hatVerwandten(1,2)
deren Vetter hatVerwandten(1,2)
<PRONOUN> Vetter hatVerwandten(1,2)
<GENITIV> Vetter hatVerwandten(1,2)
Verwandten hatVerwandten(1,2)
Verwandte <GENITIV> hatVerwandten(1,2)
Großtante hatVerwandten(1,2)

Altersgenossen hatSozialeRelation(1,2)
<NE> dem <ADELSTITEL> hatSozialeRelation(1,2)
Gästen hatSozialeRelation(1,2)
Gefährtinnen hatSozialeRelation(1,2)
Beichtvaters hatSozialeRelation(1,2)
Zimmernachbarin hatSozialeRelation(1,2)
Peiniger hatSozialeRelation(1,2)
Gast hatSozialeRelation(1,2)
Gläubigen hatSozialeRelation(1,2)
<GENITIV> Nachfolger hatSozialeRelation(1,2)
<GENITIV> Befreier hatSozialeRelation(1,2)
Mitmenschen hatSozialeRelation(1,2)
<NE> Gunst der <NE> hatSozialeRelation(1,2)
Pate hatSozialeRelation(1,2)
Mätressen hatSozialeRelation(1,2)
```

```
Schulkameraden hatSozialeRelation(1,2)
Nachfolger <ENDS-S> hatSozialeRelation(1,2)
Schwäher hatSozialeRelation(1,2)
Gefährtin hatSozialeRelation(1,2)
Gäset hatSozialeRelation(1,2)
Nachfolgerin <GENITIV> hatSozialeRelation(1,2)
Schwägerin hatSozialeRelation(1,2)

Vertraute <GENITIV> hatFreund(2,1)
Freund des <NE> hatFreund(1,2)
<NE> befreundeten <NE> hatFreund(1,2)
<NE> schließt Freundschaft mit <NE> hatFreund(1,2)
Freunden hatFreund(1,2)
Freundes hatFreund(1,2)
<GENITIV> Jugendfreund hatFreund(1,2)
Unterweltsgenossen hatFreund(1,2)
Jugendfreund hatFreund(1,2)
<NE> als Freund hatFreund(1,2)
<NE> schloß Freundin hatFreund(1,2)
Kameraden hatFreund(1,2)
<NE> geliebter Freund hatFreund(1,2)
Schulfreundin hatFreund(1,2)
<NE> , begleitet von Freund hatFreund(1,2)
Vertraute des , <NE> hatFreund(1,2)
Freund hatFreund(1,2)
<NE> vereint und <NE> hatFreund(1,2)
<NE> Verlust des Freundes hatFreund(1,2)
Kameraden <ENDS-S> hatFreund(1,2)
Lieber hatFreund(1,2)
<GENITIV> Schulfreund hatFreund(1,2)
Wohltäter hatFreund(1,2)
<NE> in Begleitung eines Freundes hatFreund(1,2)
<NE> leben verbunden mit <NE> hatFreund(1,2)
Freundin des <NE> hatFreund(1,2)
<GENITIV> Freund hatFreund(1,2)
Freundin hatFreund(1,2)
Freunden <GENITIV> hatFreund(1,2)
<NE> aufnimmt in Freundeskreis <NE> hatFreund(1,2)
konvertierten Freunden hatFreund(1,2)
<NE> folgten einige Freunde hatFreund(1,2)
Freunde hatFreund(1,2)
<NE> anfreundet mit <NE> hatFreund(1,2)
unsern Befreundten hatFreund(1,2)
<NE> einen Freund hatFreund(1,2)
Jugendfreund <ENDS-S> hatFreund(2,1)
<NE> einen Kameraden hatFreund(1,2)

<NE> heiratet <NE> hatEhepartner(1,2)
<NE> Hochzeit mit <NE> hatEhepartner(1,2)
```

```
mir gebar <NE> hatEhepartner(1,2)
<NE> kann heiraten <NE> hatEhepartner(1,2)
<NE> bis zur Heirat mit <NE> hatEhepartner(1,2)
<NE> geht und heiratet <NE> hatEhepartner(1,2)
<NE> hat und heiratet <NE> hatEhepartner(1,2)
<NE> vermählt mit <NE> hatEhepartner(1,2)
<NE> heiratete <NE> hatEhepartner(1,2)
<NE> Jawort und <NE> hatEhepartner(1,2)
<NE> geheiratet hat <NE> hatEhepartner(1,2)
<NE> Weg führt in Ehe mit <NE> hatEhepartner(1,2)
<NE> hätten gibt und heiratet <NE> hatEhepartner(1,2)

<NE> erwählt <NE> hatGeliebten(1,2)
<NE> liebt <NE> hatGeliebten(1,2)
Jugendliebe hatGeliebten(1,2)
Liebhaber hatGeliebten(1,2)
<NE> der Liebsten hatGeliebten(1,2)
Geliebte des <NE> hatGeliebten(1,2)
Freier hatGeliebten(1,2)
<NE> verliebt in <NE> hatGeliebten(1,2)
<NE> liebe <NE> hatGeliebten(1,2)
<NE> Liebschaft mit <NE> hatGeliebten(1,2)
Geliebte <NE> hatGeliebten(1,2)
Anbeter hatGeliebten(1,2)
<NE> gelingt anbetete <NE> hatGeliebten(1,2)
<NE> Liebe <NE> hatGeliebten(1,2)
Geliebte hatGeliebten(1,2)
<NE> nicht Liebschaft mit <NE> hatGeliebten(1,2)
<NE> Zuneigung zu <NE> hatGeliebten(1,2)
Liebling der <NE> hatGeliebten(1,2)
Liebe hatGeliebten(1,2)
<NE> Liebe als <NE> hatGeliebten(1,2)
<NE> Geliebte hatGeliebten(1,2)
<NE> verlebt in Liebe zu <NE> hatGeliebten(1,2)
Liebhaber <NE> hatGeliebten(1,2)
<NE> Liebe zu <NE> hatGeliebten(1,2)
<NE> in Leidenschaft zu <NE> hatGeliebten(1,2)
<NE> Beziehung zu <NE> hatGeliebten(1,2)
<NE> war verliebt in <NE> hatGeliebten(1,2)
<NE> und Geliebten hatGeliebten(1,2)
<NE> leidet unter Liebe zur <NE> hatGeliebten(1,2)
<NE> liebte <NE> hatGeliebten(1,2)
<NE> den Liebenden hatGeliebten(1,2)
<NE> lässt von Liebe zu <NE> hatGeliebten(1,2)
Liebhaber <GENITIV> hatGeliebten(1,2)
<NE> aus Liebe zur <NE> hatGeliebten(1,2)
seine Neigung zu <NE> hatGeliebten(1,2)
<NE> in verliebt <NE> hatGeliebten(1,2)
Angebetete hatGeliebten(1,2)
```

```
Geliebte <GENITIV> hatGeliebten(1,2)
geliebte <NE> hatGeliebten(1,2)
Vertrauten des <NE> hatGeliebten(1,2)
<NE> Zuneigung wird erwidert von <NE> hatGeliebten(1,2)
Geliebten <ENDS-S> hatGeliebten(1,2)
<NE> liebt und <NE> hatGeliebten(1,2)
<NE> verliebte in <NE> hatGeliebten(1,2)
<NE> bandelt mit <NE> hatGeliebten(1,2)
<NE> Liaison zu <NE> hatGeliebten(1,2)
<NE> geliebten <NE> hatGeliebten(1,2)
Geliebten hatGeliebten(1,2)

<NE_NN> <GENITIV_NN> hatRelation(1,2)
<GENITIV_NN> <NE_NN> hatRelation(1,2)
<NE> hatRelation(1,2)

<BERUF> hatBeruflicheRelation(1,2)
<GENITIV_NN> <BERUF> hatBeruflicheRelation(1,2)
<BERUF> <GENITIV_NN> hatBeruflicheRelation(1,2)
```

D.2. Ruleset for the Detection of valid Dependency paths in the OBIE algorithm

```
              # kleine, schmale Nase
CJ>>NK>>root<<
#Es war ein kleiner Mann, ungefähr fünfzig, mit einem großen Mund, einer
    scharfen Nase
CJ>>NK>>MNR>>root<<
# so rot wie sein Haar
CC>>MO>>root<<
#Ritter mit schwarzem Haar und Brandnarben
CJ>>CD>>NK>>MNR>>root<<
#schlanke Finger und Hände
CJ>>CD>>root<<NK
#schlanke Finger, Zehen und Hände
CJ>>CD>>CJ>>root<<NK
#schlanke Finger, Arme
CJ>>root<<NK
#schlanke Mutter
NK>>root<<
#großer und schlanker Vater
CJ>>CD>>NK>>root<<
#große, hübsche und schlanke Mutter
CJ>>CD>>CJ>>NK>>root<<
#Die Nase und Lippen sind blau
CJ>>CD>>SB>>root<<PD
# Die Nase ist rot und [die Hand blau] (directly attached by ommiting a verb)
```

```
PD>>root<<
#Nase, Lippen und [Hände sind grün]
PD>>root<<SB<<CJ<<CD<<CJ
#Nase, [Lippen und Hände sind grün]
PD>>root<<SB<<CJ
#Er liebt sie
SB>>root<<PD
```