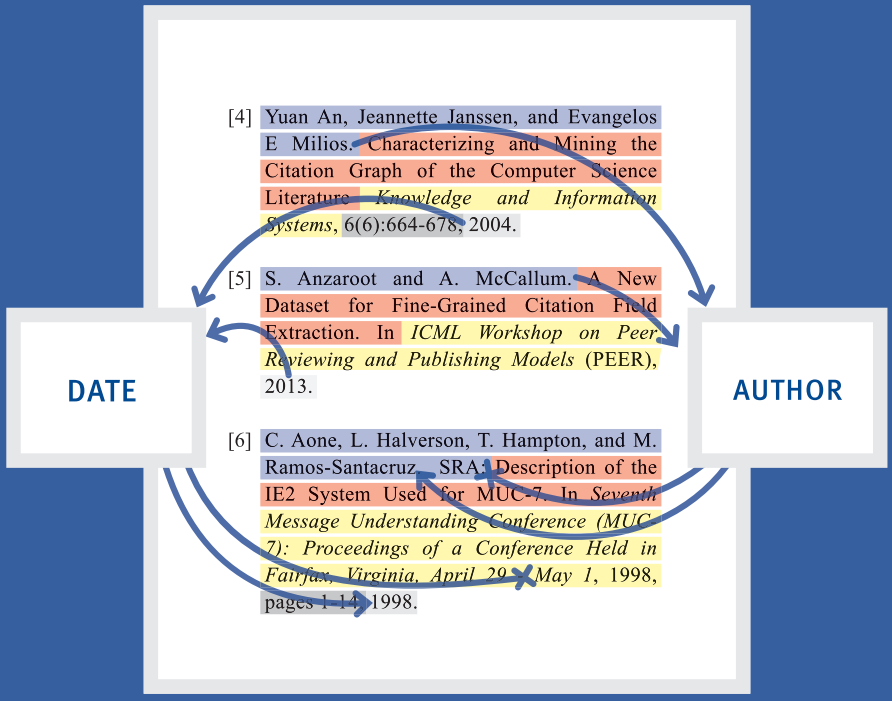
Peter Klügl

Context-specific Consistencies in Information Extraction

Rule-based and Probabilistic Approaches



Würzburg University Press

Peter Klügl

Context-specific Consistencies in Information Extraction

Peter Klügl

Context-specific Consistencies in Information Extraction

Rule-based and Probabilistic Approaches

Würzburg University Press

Dissertation, Julius-Maximilians-Universität Würzburg
Fakultät für Mathematik und Informatik, 2014
Gutachter: Prof. Dr. Frank Puppe, Prof. Dr. Prof. h.c. Andreas Dengel, Prof. Dr. Ulrich Furbach

Impressum

Julius-Maximilians-Universität Würzburg
Würzburg University Press
Universitätsbibliothek Würzburg
Am Hubland
D-97074 Würzburg
www.wup.uni-wuerzburg.de

ISBN 978-3-95826-018-4 (print)
ISBN 978-3-95826-019-1 (online)
URN urn:nbn:de:bvb:20-opus-108352

Abstract

Large amounts of communication, documentation as well as knowledge and information are stored in textual documents. Most often, these texts like webpages, books, tweets or reports are only available in an unstructured representation since they are created and interpreted by humans. In order to take advantage of this huge amount of concealed information and to include it in analytic processes, it needs to be transformed into a structured representation. Information extraction considers exactly this task. It tries to identify well-defined entities and relations in unstructured data and especially in textual documents.

Interesting entities are often consistently structured within a certain context, especially in semi-structured texts. However, their actual composition varies and is possibly inconsistent among different contexts. Information extraction models stay behind their potential and return inferior results if they do not consider these consistencies during processing. This work presents a selection of practical and novel approaches for exploiting these context-specific consistencies in information extraction tasks. The approaches direct their attention not only to one technique, but are based on handcrafted rules as well as probabilistic models.

A new rule-based system called UIMA Ruta has been developed in order to provide optimal conditions for rule engineers. This system consists of a compact rule language with a high expressiveness and strong development support. Both elements facilitate rapid development of information extraction applications and improve the general engineering experience, which reduces the necessary efforts and costs when specifying rules.

The advantages and applicability of UIMA Ruta for exploiting context-specific consistencies are illustrated in three case studies. They utilize different engineering approaches for including the consistencies in the information extraction task. Either the recall is increased by finding additional entities with similar composition, or the precision is improved by filtering inconsistent entities. Furthermore, another case study highlights how transformation-based approaches are able to correct preliminary entities using the knowledge about the occurring consistencies.

The approaches of this work based on machine learning rely on Conditional Random Fields, popular probabilistic graphical models for sequence labeling. They take advantage of a consistency model, which is automatically induced during processing the document. The approach based on stacked graphical models utilizes the learnt descriptions as feature functions that have a static meaning for the model, but change their actual function for each document. The other two models extend the graph structure with additional factors dependent on the learnt model of consistency. They include feature functions for consistent and inconsistent entities as well as for additional positions that fulfill the consistencies.

The presented approaches are evaluated in three real-world domains: segmentation of scientific references, template extraction in curricula vitae, and identification and categorization of sections in clinical discharge letters. They are able to achieve remarkable results and provide an error reduction of up to 30% compared to usually applied techniques.

Preface

Understanding natural language is a key technology for business and society, since computer services would be much more powerful if they are able to extract relevant information from free text in the web, in news, in reports and protocols etc. The last years showed impressive success for this difficult task. The book of Peter Klügl contains several innovative contributions for improving information extraction, i.e. to extract predefined information types from text like employers and durations from curricula vitae or symptoms, diagnoses and therapies from medical reports. His key idea is to exploit the observation that documents written by the same author or generated by the same process are usually much more similar to each other than to other documents of the same domain. If an information extraction system has extracted information from one document of such a cluster, context specific features of the extracted information can be learned and transferred for extraction from other documents of the same cluster. Peter Klügl shows convincingly, that this method of exploiting context specific consistencies in documents works both for machine learning and rule based information extraction approaches and improves precision and recall in several applications with publicly available corpora and own projects by about 30%.

Information extraction is a difficult engineering task: researchers and practitioners use a lot of tools, often integrated in architectures like GATE (general architecture for text engineering) or UIMA (unstructured information management architecture). Peter Klügl has designed and implemented an innovative open-source rule-based component UIMA Ruta (rule-based text annotation) for UIMA thus closing a gap for this widespread architecture. His book is characterized by both theoretical and practical contributions and insights for information extraction and I recommend it strongly.

Frank Puppe
Chair VI – Artificial Intelligence and Applied Computer Science
Institute of Computer Science
University of Würzburg

Acknowledgements

This work would not have been possible without the help and support of many people.

First of all, I want to thank my supervisor Frank Puppe for his guidance, expertise and feedback. He provided a great and enjoyable research environment and he always made time for some open-minded discussions and research questions. I really enjoyed the time in his working group. Furthermore, I like to thank Andreas Hotho, Joachim Baumeister and my sister Franziska Klügl-Frohnmeyer for their support. They greatly influenced my interest in research in general and especially in artificial intelligence.

Many thanks to my colleagues and friends at our working group for their help concerning technical, organizational or scientific problems: Alexander Hörnlein, Florian Lemmerich, Georg Dietrich, Georg Fette, Jochen Reutelshöfer, Marianus Ifland, Maximilian Ertl, Markus Krug, Petra Braun, Philip-Daniel Beck, Reinhard Hatko and all others. A special thank goes to Martin Toepfer, who helped me to push my research forward and assisted me in the creation of various probabilistic models as well as UIMA Ruta. Also many students, especially Andreas Wittek, Benjamin Eckstein and Tobias Hermann, have done their bit in the development of UIMA Ruta.

Finally, I want to thank my parents and Alrun Eiche for their understanding and ongoing support during all these wonderful years.

Würzburg, August 2014 / January 2015

Contents

Chapter 1

Introduction

Information extraction addresses the identification of well-defined entities and relations in unstructured data and especially in textual documents. Even if the research in this area has a long history, it becomes more and more important nowadays due to the increased availability of unstructured data. A vast amount of information is stored and exchanged in an unstructured representation since it is mainly intended to be interpreted by humans. Examples for this fact are webpages in the Internet, technical documents in industry, medical notes or even novels[1].

In order to access the concealed information for analytic processes, it has to be transformed into a structured representation. Hence, information extraction has become a key component in the integration of textual data and can be considered as an umbrella term for many interesting tasks such as named entity recognition, sentiment analysis or knowledge extraction. Two emerging and challenging trends in this area are information extraction from social media, like blogs or tweets [162], and knowledge extraction from clinical notes [179].

Approaches to information extraction can roughly be divided into two main categories: approaches based on handcrafted rules and approaches based on statistical models trained in a supervised fashion. The latter models include classifiers or probabilistic graphical models like Conditional Random Fields [137]. There are of course no clear boundaries since hybrid information extraction systems can apply components of both approaches, or the rules are not written by a knowledge engineer, but they are automatically induced. While statistical models dominate the research in academia, commercial applications are mostly implemented as rule-based systems [44]. This discrepancy cannot be explained by the latency of translational efforts from research to industry. Chiticariu et al. have investigated the reasons for this disconnect and noticed that research and industry measure the costs and benefits of information extraction differently [44]. Aside from many other reasons, rule-based systems sometimes fit better in the requirements of real-world use cases, e.g., availability of labeled data, stability of the specification or traceability of results. Statistical models are able to include a vast amount of properties and features in the classification decision, which often leads to an improved accuracy compared to rule-based approaches. It is easier to publish approaches using machine learning techniques since no human factor needs to be evaluated in the experimental setup and the experiments can be reproduced given the algorithms and the labeled data. Even though statistical models often perform better, e.g., in information extraction challenges, the need for rule-based information extraction will not decrease in the foreseeable future. The amount of publications about rule-based systems and approaches has even been slightly increasing in the recent years [66].

[1] Parts of the content of the introduction are taken from Peter Kluegl, Martin Toepfer, Philip-Daniel Beck, Georg Fette, Frank Puppe. UIMA Ruta: Rapid Development of Rule-based Information Extraction Applications. Natural Language Engineering, 2014 [125].

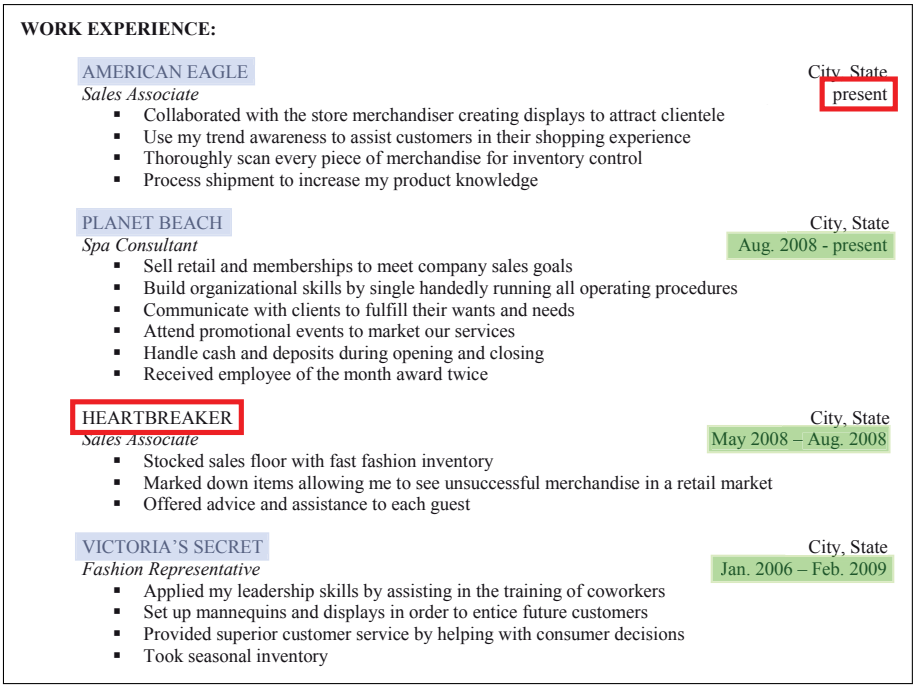
1.1 Motivation

When extracting entities and relations it is often assumed that they are independent and identically distributed (iid). However, many documents violate these assumptions. Webpages and semi-structured documents are often generated using templates that specify the arrangement of the textual contents. The entities of a specific type of information are ordered in lists, tables or similar structures and share the same layout. Examples for this are web stores that provide one webpage for each product. The information about the product is arranged with the same layout and formatting in each page. The title of the product may use a different layout than the price, but these entities are consistently structured throughout all pages of a website. Other websites apply, however, different templates for generating the webpages. While the entities in this website are also consistently structured, they might apply different layouts compared to the previous website. When an information extraction model processes these websites, for example, in order to extract the product information, then the iid assumption is violated. The entities are not independently distributed since subsets of entities share similar layouts.

Such dependencies between entities do not only occur in automatically generated documents. When humans manually compose and write documents, then they typically apply some sort of template or consistent structure. They organize repeating entities homogeneously or keep the order of entities. Humans tend to apply the same layout for elements of equal level, like the fonts of different kinds of headlines. These regularities can especially be found in semi-structured documents that include layout or structure in order to highlight different aspects of the text.

This work calls the dependencies between entities context-specific consistencies. The entities within one context or document share the same or at least similar composition, which manifests in the choice of formatting, the order of entities or other properties. Entities in another document are also consistently structured, but in a different way. The actual composition of the entities is not known when processing a document, and the composition and applied formatting is potentially contradictory. These consistencies concerning the arrangement and layout of entities are called context-specific, because they are only fulfilled in a specific frame or scope. Information extraction models processing these kinds of documents face severe problems resulting in a reduced performance. Properties that indicate a specific type of entity in one document are not valid or give evidence for a different type of entity in another document. The discriminative effect of these properties thus stays behind its potential. If these properties can be utilized restricted in a certain context or document, then they enable the information extraction model to considerably improve their results. When humans read documents with context-specific consistencies, they are actually able to easily interpret the contained information. They automatically consider the context and include similarities between entities for identifying the information.

This problem is illustrated with an example of extracting entities from curricula vitae. Figure 1.1 depicts a fictional curriculum vitae and highlights the results of an information extraction model, e.g., a set of rules built or statistical model with dictionary lookup. If the applied dictionaries contain entries for *American Eagle*, *Planet Beach* and *Victoria's Secret*, then these companies can be easily identified without further keywords. However, the company *Heartbreaker* will probably not be extracted since not enough evidence is given due to the missing entry in the dictionary. The model was able to identify common patterns of timespans but failed to detect the first occurrence "present", which indicated the current position of the author.



WORK EXPERIENCE:

AMERICAN EAGLE — City, State
Sales Associate — present
- Collaborated with the store merchandiser creating displays to attract clientele
- Use my trend awareness to assist customers in their shopping experience
- Thoroughly scan every piece of merchandise for inventory control
- Process shipment to increase my product knowledge

PLANET BEACH — City, State
Spa Consultant — Aug. 2008 - present
- Sell retail and memberships to meet company sales goals
- Build organizational skills by single handedly running all operating procedures
- Communicate with clients to fulfill their wants and needs
- Attend promotional events to market our services
- Handle cash and deposits during opening and closing
- Received employee of the month award twice

HEARTBREAKER — City, State
Sales Associate — May 2008 – Aug. 2008
- Stocked sales floor with fast fashion inventory
- Marked down items allowing me to see unsuccessful merchandise in a retail market
- Offered advice and assistance to each guest

VICTORIA'S SECRET — City, State
Fashion Representative — Jan. 2006 – Feb. 2009
- Applied my leadership skills by assisting in the training of coworkers
- Set up mannequins and displays in order to entice future customers
- Provided superior customer service by helping with consumer decisions
- Took seasonal inventory

Figure 1.1: Excerpt of a fictional curricula vitae[2] with highlighted spans for *Companies* and *Dates*. The boxed areas exemplify missing entities caused by the absence of specific features. The remaining contextual features are not sufficient for identifying these entities. Exploiting the context-specific consistencies concerning positioning and formatting can help to find the missing entities.

In this example, the missing company can simply be identified by analyzing the other companies in this curriculum vitae. They all contain only capitalized characters and are located at the beginning of the section describing an employment. The same applies for the missing timespan of the first employment. All extracted timespans are located in the second line after the title of the job position. By identifying and using the knowledge about the consistent composition of the entities in these sections, many errors of the extraction model can potentially be prevented. Especially precision-driven approaches based on dictionaries and rules can be improved by increasing their recall. However, also statistical models are able to take advantage of context-specific hints. They exploit the consistent composition of entities in order to compensate missing features or contradictory formatting. The example straightforwardly highlights the weaknesses of information extraction models compared to humans. While information extraction models typically process the document using only local features, humans take advantage of evidence that is interspersed throughout the complete text. They always incorporate the context and automatically infer a model about the typical composition. As a consequence, the information is easily identified.

[2] `http://upload.wikimedia.org/wikipedia/commons/c/cc/Resume.pdf`

1.2 Goal

The main claim of this work follows logically from the motivation in the last section: context-specific consistencies can help to improve information extraction. Some sparse and isolated publications have already considered these kinds of dependencies [25, 177, 94, 92]. However, their approaches are limited concerning isolated factors. Some of them have been developed for a specific task and are not applicable for other domains. Most publications present approaches based on one specific technique. Furthermore, all approaches are limited concerning the kinds of consistencies they are able to exploit. There is no common understanding of the problem, which led to redevelopment of the same ideas. A thorough and systematic investigation of context-specific consistencies in information extraction is necessary that is not limited to specific techniques, domains or aspects of consistencies.

The goal of this work consists in this investigation of context-specific consistencies. A closer look has to be taken at possible appearances of context-specific consistencies and approaches need be developed that rely on different techniques and that are not limited to a specific domain. The focus lies on practical solutions instead of models of arbitrary complexity. Approaches are preferred that extend available techniques while providing considerable advantages over standard solutions. Furthermore, the effort of developing information extraction models with techniques for exploiting context-specific consistencies should not exceed the benefit, e.g., the increase of accuracy.

The two main directions for developing information extraction models consisting in handcrafting rules and training probabilistic models provide different advantages and disadvantages. Their applicability for an information extraction task depends on various factors. These include the availability of examples, the effort to create gold standards, the necessity of traceable results, the stability of the specification, or the required performance including runtime and quality. The occurrence of context-specific consistencies is independent of these factors. Thus, approaches need to be developed that are able to exploit the consistencies using handcrafted rules as well as trained models.

The development of information extraction methods and applications is not an end in itself. They are created to solve or improve specific information extraction tasks in given domains. The generality and applicability of the developed approaches are shown in three domains: references in scientific publication, curricula vitae and clinical discharge letters. These domains are very different concerning various characteristics like the information extraction task and highlight the widespread occurrences of context-specific consistencies.

1.3 Contributions

The main contribution is the systematic investigation of context-specific consistencies in the research area of information extraction. This work provides the first thorough analysis of this kind of dependencies, which extends the effectiveness of information extraction in many domains. Some sparse and isolated publications present patchwork restricted to certain techniques, domains and the kinds of consistencies they are able to model. The results of this work are not limited concerning these factors. They provide solutions for exploiting various kinds of context-specific consistencies. Most of these consistencies exceed the capabilities of related

approaches. This work is not restricted to a specific technique for information extraction. It presents solutions how to take advantage of the consistencies with the two major approaches in this area: handcrafted rules and probabilistic models. Thus, the presented approaches are able to fit in any kind of requirements of real-world use cases. The general applicability of the developed approaches is shown by evaluating them in three dissimilar domains, in which the accuracy of the information extraction task could be greatly increased.

The technical contribution of this work to the research field of information extraction consists of four major parts:

- Design and implementation of the rule-based system UIMA Ruta.
- Knowledge engineering approaches for exploiting context-specific consistencies in rule-based applications.
- Learning a model of context-specific consistencies with rule-based classifiers.
- Extensions of Conditional Random Fields for exploiting context-specific consistencies in information extraction based on machine learning.

This section provides an overview of the different parts of contribution and reports the experimental results in three real-world domains: segmentation of references, template extraction in curricula vitae and segmentation of clinical letters. These three domains are very different concerning the information extraction task, which highlights the widespread occurrences of context-specific consistencies and the general applicability of the developed approaches.

UIMA Ruta

UIMA Ruta is a rule-based system for information extraction and general natural language processing tasks. A special focus of the system lies on a compact rule language with a high expressiveness combined with strong development support. These attributes facilitate rapid development of rule-based applications and thus reduce one major bottleneck when handcrafting extraction knowledge: the time and costs of the engineering task. UIMA Ruta provides most of the features of related systems concerning language and tooling support. Furthermore, it introduces several new and useful elements that are not found in other systems. These include amongst others a coverage-based concept of visibility, powerful vertical matching or estimation of the rules' quality on unlabeled documents.

Rule-based information extraction is a well-established field of research with a long history and various systems and implementations. There are, however, several reasons for the development of a new system. Besides the provided features in contrast to related work, the support of rule-based systems in the UIMA framework [77] lacks freely available, open source alternatives. Commercial or partially commercial systems are not straightforwardly utilized in academic context. Other rule languages based on UIMA lack expressiveness or tooling support. The absence of a freely available and open system that is accepted as the standard tool by the community handicaps academic research. This fact can be substantiated with examples from other architectures like JAPE [54] in the GATE framework [58]. UIMA Ruta tries to fill this gap of standard rule support for the UIMA framework. The success can hardly be measured, but

there are already strong indicators that the system establishes in the community, e.g., questions asked in mailing lists and in websites like Stack Overflow[3].

UIMA Ruta has not been created specifically for exploiting context-specific consistencies. It is a useful general-purpose tool for many diverse use cases. In contrast to other systems, however, it provides some features that facilitate the integration of context-specific consistencies in rule-based applications. Amongst others, these include language elements rather unknown to rule languages. Lists and variables help to model consistencies with rules and allow one to integrate dynamic knowledge about the currently processed document. An example for this fact is the usage of variables in the matching condition of a rule in order to process an entity dependent on the dominant composition of all entities. Another important aspects consists in the availability of different engineering approaches. Due to the high expressiveness of its language, UIMA Ruta supports always more than one approach to solve an annotation problem, which can be essential for dealing with different aspects of consistencies. Besides these properties, efficient and effective engineering is also helpful when handcrafting rule set for context-specific consistencies and leads to an improved engineering experience in general.

Knowledge Engineering Approaches

The first contribution for exploiting context-specific consistencies in order to improve the performance of information extraction is a set of case studies that have been implemented using UIMA Ruta. These rule-based applications highlight different engineering approaches that enable a knowledge engineer to take advantage of the consistencies.

In the first case study, simple precision-driven rules are extended with rules for finding additional entities that share a composition similar to detected ones. This approach is very effective and efficient since the utilized rules are easy and fast to engineer. Even if the resulting applications are maybe not sufficient for solving real-world information extraction tasks, they provide many advantages. The approach can be utilized for creating prototypes, which perform almost as good as throughout engineered applications, or for a fast analysis of a domain, e.g., by extracting lists of potential entities. The results can then be applied to accelerate the development of the actual application. The engineering approach is evaluated for the identification of companies in curricula vitae and for detecting the existence of headlines in clinical discharge letters. Both rule sets are able to achieve an F_1 score of over 0.97 for unseen documents in the respective domain and have been created in less than two hours. The experimental setup is limited but reflects realistic conditions in real-world scenarios. A feasibility study should be performed with a limited amount of documents, which prevents the usage of supervised machine learning approaches.

The second case study considers a different domain and a more sophisticated approach for exploiting context-specific consistencies. Here, rules are applied in a transformation-based manner in order to segment scientific references. An initial set of rules extracts interesting entities like author, title and date. Then, rules investigate the composition of the entities and create a model of different aspects of the occurring context-specific consistencies. This model is utilized by transformation-based rules in order to modify the initial entities. The rules reclassify and change the offsets of the entities until they confirm with the dominant composition of entities

[3] `http://stackoverflow.com/questions/tagged/ruta`

of the same type in the current document. The procedure greatly increases the accuracy of the application if the assumption about consistent composition of entities is fulfilled. This case study illustrates some advantages of the rule language of UIMA Ruta. The model of consistencies is specified using only language elements of UIMA Ruta. The properties of the consistent composition are stored in variables and thus can be directly integrated in rules. Furthermore, the necessary transformations can be efficiently specified in UIMA Ruta since the language allows one to modify arbitrary annotations with specialized actions. The combination of the three phases is able to achieve an F_1 score of 0.997 for consistently formatted references. While the experimental setup is limited, it highlights the power of this engineering approach. The three phases do not need to be implemented using rules, but can also rely on arbitrary techniques. The initial extraction of entities may rely statistical models and the analysis of the consistencies on proprietary algorithms. The last phase consisting of the transformation-based knowledge is already well-suited.

The last case study investigates the usage of consistencies in a complete application. The rules of this case study identify and categorize sections in clinical discharge letters. Overall, 48 categories including one category for other kinds of sections are distinguished. In the sections, specialized information extraction models are applied in order to populate a clinical data warehouse. The rules utilize different features including semantic keywords, formatting and also context-specific consistencies in order to remove false positive headlines in a set of potential candidates. Thus, the consistencies are applied in order to increase the precision. Additional rules utilize the headlines in order to identify the sections and their categories. The rules are able to achieve an F_1 score of 0.992 for the extraction of the headlines in a test set of 200 unseen documents. Overall, this case study highlights the usefulness of exploiting context-specific consistencies in a throughout engineered application. Although the rules have been optimized to perform in a best possible manner, the integration of the consistencies was still able to further improve the accuracy.

Learning Context-specific Consistencies

The first contribution based on machine learning techniques does not concern improvements of an information extraction model, but the induction of a model for the context-specific consistencies, which occur in the currently processed document. The general approach consists in utilizing binary classifiers in order to describe specific aspects of the consistencies like the boundaries of entities or transition between two specific types of entities. This approach provides several advantages compared to related work. Learning a model represented by a set of classifiers enables the usage of a combination of features instead of only one specific property. Thus, the consistencies can be described more precisely. Another advantage of a distinct model consists in the fact that this model can be applied not only to make statements about how consistent an entity is, but also what other, not yet considered positions fulfill the different aspects of consistency.

The general procedure for learning the context-specific consistencies can be described the following way. First, an initial prediction of entities in a document needs to be provided. These entities together with a subset of available features built a new dataset for the currently processed document. The classification model is trained to classify specific aspects of the entities of one type like their beginning. After a model is acquired, it is applied on the same dataset

in order to provide evidence about consistent and inconsistent positions. The capability to generalize over the given examples leads to a classification of the correct positions. There are four important properties that the classifier needs to fulfill. It should not tend to overfit on the data, its hypotheses space need to confirm with the assumption, it should handle label bias correctly, and it should be fast since it is learnt and applied during processing a document.

In general all kind of classifiers can be applied in order to learn a model of the context-specific consistencies. This work utilizes subgroup discovery in order to learn simple binary classification rules that consist of a conjunction of features. Subgroup discovery provides several advantages over popular classifiers like Support Vector Machines. The learnt subgroups or rules can be easily interpreted by humans, which facilitates the development process. Furthermore, subgroup discovery straightforwardly supports the usage of proprietary quality functions. This feature is exploited in the context of this work by integrating a novel quality function that takes advantage of additional background knowledge, i.e. the expected amount of entities of one type. Using this improvement, correct models of context-specific consistencies can already be learnt with a minimal amount of given entities. It is able to greatly increase the correctness of the predicted positions.

The modeling of the context-specific consistencies is evaluated in two experimental settings separately from the actual information extraction task. In both settings, the domains of curricula vitae and references are considered. The first experiment investigates the ability of subgroup discovery to classify the correct boundaries of entities given erroneous training data. For this purpose, a gold standard dataset is incrementally deteriorated by replacing a correct boundary by an incorrect one. The learnt rules are able to greatly improve the F_1 score and even classify the boundaries of the entities correctly despite of only minimal amount of correct input data. The second experiment investigates the performance of subgroup discovery on more realistic predictions. The learnt rules achieve an error reduction of 16%-30% for the classification of boundaries compared to the prediction provided by a five-fold Conditional Random Field.

Machine Learning Approaches

The last contribution concerns the usage of the learnt model of context-specific consistencies in Conditional Random Fields in order to improve the information extraction result. The approaches and models developed in the context of this work utilize available implementations in contrast to the rule-based approaches, which rely on UIMA Ruta. Machine learning approaches and especially Conditional Random Fields are very popular in information extraction and thus a variety of suitable implementations are available.

Overall, three approaches based on Conditional Random Fields are presented. The first approach applies stacked Conditional Random Fields with two phases. The initial model provides a prediction of the entities in a document, which is utilized in order to learn a model of the consistencies. The stacked model is extended with additional feature functions that provide static semantics for the model, but change their manifestation dependent on the currently induced consistencies. This is achieved by directly utilizing the rules of the consistency model as feature functions. The approach is evaluated for the segmentation of references and is able to achieve an F_1 score of 0.940 in a cross fold setting. This constitutes an error reduction of 30% compared to a baseline Conditional Random Field.

The remaining approaches based on Conditional Random Fields add additional factors dependent on the learnt consistency model. Both models utilize a prediction of entities provided by an external Conditional Random Field. Thus, their approach is similar to stacking the models, but they potentially also support an incremental unfolding of the graph during inference. The first model called comb-chain extends the probabilistic graph with unigram factors on the positions indicated by the learnt rules. New feature functions represent consistent positions (true positive) as well as missing (false positive) and additional (false negative) positions compared to the prediction. The second model is called skyp-chain and extends the graph structures with long-range dependencies between interesting positions. The same set of feature functions is applied as before. The experimental results for the segmentation of references and entity extraction in curricula vitae again indicate an error reduction of up to 30% compared to a baseline Conditional Random Field. The comb-chain model overall performs better due to inference problems in the more complex graph structure of the skyp-chain model. It achieves an F_1 score of 0.976 (references), 0.962 (dates in curricula vitae) and 0.814 (companies in curricula vitae). The discrepancy of the results concerning the segmentation of references compared to the results of the stacked approach is caused by the usage of a different set of labels, by the utilized implementation and by the differing specification of an instance.

1.4 Structure of this Work

This chapter provided an introduction to the topic "Context-specific Consistencies in Information Extraction". The utilization of this special kind of consistencies in order to improve information extraction models is motivated and outlined. Furthermore, the contribution to the field of information extraction is summarized. The remainder of this work is structured in six chapters. The related work of this work's contributions is described in Chapter 2 and Chapter 3.

Chapter 2 contains an overview of information extraction in general. It starts with the foundations that include a categorization of the task, the historical development, typical evaluation measures and established architectures. The rest of the chapter describes the two major directions in information extraction: handcrafted rules and models based on machine learning. The description of rule-based information extraction is more detailed since it represents the related work of a contribution of this work that is the rule-based system UIMA Ruta. The last section provides an introduction to information extraction based on machine learning approaches. It describes the essentials and the application for information extraction. Additionally, a description of Conditional Random Fields introduces the fundamental basics for the contributions of this work concerning machine learning approaches.

Chapter 3 represents the core of this work. It provides an introduction to context-specific consistencies and investigates their general characteristics. The concept is illustrated with three exemplary domains with context-specific consistencies: references of scientific publications, curricula vitae and clinical discharge letters. These domains are utilized for the experimental evaluations of the knowledge engineering and machine learning approaches of this work. For each domain, the targeted information extraction task is defined and interesting applications are outlined. Furthermore, related work in that domain and especially for the targeted task is summarized. The section of each domain concludes with a description of the context-specific consistencies in this domain and how they influence information extraction. A short out-

look summarizes further domains where context-specific consistencies occur. The last section considers related work concerning context-specific consistencies and the usage of long-range dependencies in collective information extraction.

Chapter 4 contains the description of the rule-based system UIMA Ruta, which has been developed in the context of this work. The chapter starts with a short introduction and the historical development. Then, the rule-based scripting language is described in detail by elaborating on syntax and semantics, inference, novel language elements, and different engineering approaches. Then, the development environment and tooling of UIMA Ruta is presented, which covers basic development support, explanation of rule execution, introspection by querying, automatic validation, constraint-driven evaluation, supervised rule induction, and semi-automatic creation of gold documents. The chapter concludes with a comparison to related systems, which especially illustrates the compactness and expressiveness of the language.

Chapter 5 describes three case studies of rule-based approaches for exploiting context-specific consistencies. Each case study has been implemented using UIMA Ruta. The first case study applies precision-driven rules, which are extended by rules sensitive to the consistencies for increasing the recall. The developed rule sets represent prototypes that process curricula vitae and clinical letters. The second case study applies transformation-based rules for segmenting scientific references and the third case study highlights the usefulness in a complete application for segmenting clinical letters. The chapter concludes with a discussion of the approaches.

Chapter 6 considers the contributions of this work concerning machine learning approaches. It starts with a description how a model of context-specific consistencies can be learnt. An example and an experimental study illustrate the power and advantages of the developed approach. The remainder of the chapter provides descriptions and evaluations of three models based on Conditional Random Fields that utilize the learnt consistencies in order to improve the extraction result. The experimental evaluation considers only the domains references and curricula vitae since the rule-based approaches covered the clinical letters to an extend where hardly any improvement can be accomplished. The chapter concludes with a discussion of the contributions.

Finally, Chapter 7 concludes the work with a summary and an outlook on different possibilities for future work.

Chapter 2

Information Extraction

Information extraction (i.a. [124, 196, 162, 176]) addresses the identification of well-defined entities and relations in unstructured data and especially in textual documents. Even if the research in this task has a long history, it becomes more and more important nowadays due to the increased availability of unstructured data. In order to access the concealed information for analytic processes, it has to be transformed into a structured representation. Hence, information extraction has become a key component in the integration of textual data and can be considered as an umbrella term for many interesting tasks such as named entity recognition, sentiment analysis or knowledge extraction[4].

Approaches to information extraction can roughly be divided into two main categories: approaches based on handcrafted rules and approaches based on statistical models trained in a supervised fashion. The latter models include classifiers or probabilistic graphical models like Conditional Random Fields [137]. There are of course no clear boundaries since hybrid information extraction systems can apply components of both approaches, or the rules are not written by a knowledge engineer, but they are automatically induced. While statistical models dominate the research in academia, commercial applications are mostly implemented as rule-based systems [44]. This discrepancy cannot be explained by the latency of translational efforts from research to industry. Chiticariu et al. have investigated the reasons for this disconnect and noticed that research and industry measure the costs and benefits of information extraction differently [44]. Aside from many other reasons, rule-based systems sometimes fit better in the requirements of real-world use cases, e.g., availability of labeled data, stability of the specification or traceability of results. Even though statistical models often perform better in information extraction challenges, the need for rule-based information extraction will not decrease in the foreseeable future. The amount of publications about rule-based systems and approaches has even been slightly increasing in the recent years [66].

This chapter gives not an overview of the current State-of-the-Art in information extraction, but provides a description of the basics and preliminaries for the contribution of this work. First, Section 2.1 introduces information extraction together with its historical development, evaluation measures and architectures. Section 2.2 provides a description of rule-based information extraction. Different rule languages are investigated in detail. The reason for this focus on rule-based system can be found in the contribution of this work. While the techniques for the machine learning approaches rely on available implementations, the rule-based approaches build upon a newly created system. Thus, the description of the rule languages can be considered

[4] Parts of the content of the introduction are taken from Peter Kluegl, Martin Toepfer, Philip-Daniel Beck, Georg Fette, Frank Puppe. UIMA Ruta: Rapid Development of Rule-based Information Extraction Applications. Natural Language Engineering, 2014 [125].

related work for a basic part of the contribution. Information about development support for this engineering task and different approaches for rule induction conclude this section. Section 2.3 provides an introduction in the preliminaries for the machine learning approaches. First, the essentials for supervised machine learning are presented. Then, different representations of information extraction as machine learning tasks are described. The section concludes with a more detailed introduction to Conditional Random Fields, which are later applied for the machine learning approaches of this work.

2.1 Foundations

This sections gives a short introduction in the basics and foundations of information extraction. First, the task of information extraction is specified and classified according to different fields of research. The description of the historical development provides insights in the driving forces of information extraction. Then, different options of assessment for information extraction approaches in experimental evaluations are described. Finally, the last section provides a short introduction in architectures for information extraction.

2.1.1 Definition

Information extraction is the task of identifying well-defined entities and relations in unstructured data. This means that the developer of the information extraction model already knows the type of information that needs to be extracted. Given a set of documents that serves as input for the information extraction process, the output of this process consists in a set of spans referring to specific positions in the documents and a classification of the spans according the predefined types of information. Variants of information extraction allow also a broader definition of the task. Open information extraction [73, 61, 84, 205, 74, 38], for example, jointly induces types of entities and relations independently of the domain.

Information extraction is a wide field of research and can be associated with several disciplines:

- Natural Language Processing
- Text Mining
- Database research
- Computational Linguistics
- Machine Learning

Information extraction can be considered a subfield of natural language processing as it processes textual documents in order to find mentions of specific information. Text mining is a subfield of data mining. It describes the property of information extraction for mining specific entities from unstructured data. The results of information extraction are often stored in databases. Hence, research in this area is interested in information extraction in order to populate databases. Computational linguistics is related to information extraction because of the deeper analysis of the text in order to identify interesting entities. Finally, machine

learning relates to information extraction, because many information extraction approaches are based on machine learning techniques. Publications about novel models are often evaluated for information extraction tasks.

The definition of information extraction includes many different types of tasks and variants [162]:

Named Entity Recognition Named entity recognition considers the identification of named entities like persons, organizations, locations and dates. Hence, it is a limitation of information extraction for a specific subset of entities.

Coreference Resolution Coreference resolution considers the assignment of different mentions to the same entity. It includes abbreviations, pronouns, indirect and implicit mentions.

Relation Extraction This task addresses the identification of the relations between entities. The set of potential relations is typically well-defined like the types of entities.

Template Filling Template filling and event extraction consider a more complex task than relation extraction. Here, several slots for different parts of a relation need to be filled with entities.

Sentiment Analysis This task addresses the identification of subjective information and especially the polarity of statements about entities.

Ontology Population In the context of ontologies, information extraction is often called ontology population. The goal of this task consists in creating new instances for specific concepts from unstructured data.

Information extraction must not be mistaken with information retrieval. The result of the latter task is a ranked list of relevant documents that have been collected using a set of keywords. Information extraction typically is applied on a fixed set of documents and returns specific text fragments that represent mentions of specific entities. Information extraction and information retrieval often take advantage of their combination. Information retrieval can be applied in order to acquire the documents, in which information extraction should be performed. Vice versa, information extraction can be utilized in order to improve the ranking of the retrieved documents.

2.1.2 Historical Development

The research in the area of information extraction has been influenced by various factors. These include challenges, conferences, projects and funding, but also emerging domains and tasks as well as new techniques and approaches. Most survey articles and book chapters about information extraction provide a comprehensive overview of the historical development (cf. [196, 162, 176, 76]).

One major stimulus for the research in the area of information extraction have been the message understanding conferences (MUC [90]) initiated by the US Navy and sponsored by

the United States Advanced Research Projects Agency (DARPA[5]). Overall, seven conferences from 1987 to 1998 organized different challenges for information extraction tasks (cf. [196]). The first MUC in 1987 utilized naval tactical operations as the domain of the documents, but neither the extraction tasks nor evaluation measures have been declared. The second MUC in 1989 considered the same domain and additionally specified the task that should be performed. The participants should fill a template for each event whereas the slots cover aspects like type of event, agent, time and place and effect (cf. [196]). The resulting systems have been hardly comparable since the evaluation was performed by the participants themselves. MUC-3 in 1991 changed the domain to the terrorism events in Latin America. Additionally, the organizers provided training and test sets for an extended template, and they specified four evaluation measures that include precision and recall. At this time, the TIPSTER Text program[6] of DARPA was initiated in order to finance several participants in this research area. The MUC in 1992 utilized a further increased amount of slots and propagated the usage of the F measure for the evaluation. In 1993, MUC-5 introduced Japanese documents and two new domains that covered financial news and microelectronic products. The financial domain and new subtasks has been in the focus of the sixth MUC in 1995. The tasks covered named entity recognition, template filling, coreference resolution, word sense disambiguation and predicate argument syntactic structuring. In 1998, the MUC-7 included documents about airline crashes in different languages. A new task dealt with the relations between entities. After the MUC conferences, DARPA funded the TIDES program, which led to the automatic content extraction (ACE[7]) evaluations. These challenges included different tasks like entity detection, coreference resolution and relation detection with more level of types and mentions.

Challenges and projects have not only been funded by the DAPRA, but also by the European Commission (cf. [196]). First, the Linguistic Research and Engineering program supported the development of tools and components for information extraction in parallel to the MUC challenges. Later, the Pascal Network of Excellence[8] and the Dot.Kom European project[9] launched different challenges for machine learning approaches to information extraction. Other challenges have been organized by the Conference on Computational Natural Language Learning [193] and the Text Analysis Conference[10].

The historical development in the area of information extraction can also be approached from different perspectives than challenges and funding. One perspective considers the applied methods and approaches for information extraction. Until the late 1980s, handcrafted system based on rules and patterns dominated information extraction. Even when first approaches based on machine learning have been published, they were not able to compete with the engineered systems. The fact that systems like FASTUS [98] and IE2 [5] are counted as some of the best system in the MUC challenges may serve as an example for this situation. In the late 1990s, machine learning approaches won ground. Approaches based on naive Bayes classifiers and Hidden Markov Models have been published. With the emergence of vast amounts of websites, several approaches for rule induction and wrapper generation have been published.

[5] http://www.darpa.mil
[6] http://www.itl.nist.gov/iaui/894.02/related_projects/tipster/
[7] http://www.itl.nist.gov/iad/mig/tests/ace/
[8] http://www.pascal-network.org/
[9] http://nlp.shef.ac.uk/dot.kom/
[10] http://www.nist.gov/tac/

These systems learn rules and patterns in a supervised fashion in order to extract specific entities from structured and semi-structured documents. After 2000, black-box models based on statistical classifiers and probabilistic graphical models started to dominate academic research. More and more generative and especially discriminative models like Conditional Random Fields [137] have been applied in order to solve information extraction tasks. Nowadays, the amount of publications about rule-based approaches increases again slightly, but most publications in this research area propose black-box models based on machine learning.

2.1.3 Evaluation Measures

Information extraction approaches and models are typically evaluated and their performance is estimated using gold standard corpora. These documents specify the input as well as the output of an information extraction task. They consist of the unstructured and structured information, which is achieved by bundling the input text with additional annotations that represent the output. Normally, annotations for the interesting entities and relation specify the correct output of the task. The annotated documents are typically not utilized during the development of the extraction models and represent a new and unseen sample of the targeted domain. The quality of information extraction models is estimated by applying them on the raw texts of the gold standard. The output of the models is then compared to the given annotations for the documents. This comparison results in the four values known from binary classification:

True Positives (tp) The amount of correctly identified information. The gold standard information and the extracted information of the model agree for the identified positions. The information extraction model was able to reproduce the correct entities or relations.

True Negatives (tn) The amount of information that was correctly not identified. The gold standard information and the extracted information of the model agree for uninteresting positions. This value is typically not calculated for information extraction tasks since it potentially consists in a vast amount of possible spans in the document.

False Positives (fp) The amount of falsely identified information. The information extraction model identified entities or relations that are not present in the gold standard.

False Negatives (fn) The amount of missing information. The information extraction model was not able to identify entities and relations that are present in the gold standard.

The values represent the errors and correct extractions in a generic way, but the manner how they are calculated can differ. In information extraction, these values are typically given on token-level or on entity-level. When using the token-level, each token is considered separately. If a token is part of an entity of the correct type in the output of the model as well as in the gold standard, then the token is counted as one true positive. The calculation using the entity-level, the complete entities are considered. An entity is only counted as a true positive only if the offsets and the type of the entity is correct. False positives and false negatives are calculated correspondingly. Evaluations based on entity-level are more precise as a consequence, but evaluations using the token-level provide more information since also partial matches are considered. An entity of ten tokens whereas only one token at the end is not correct, for example,

produces a false positive and a false negative for the entity-level, but it results in nine true positives and one false positive using the token-level.

The values for true positives, false positives and false negatives are utilized in evaluation measures in order to estimate the quality of the model. The typical measures are the accuracy and the F measure. Most often, the F_1 measure is applied, which is defined as the harmonic mean of precision and recall. These measures are defined in the following equations:

$$accuracy = \frac{tp + tn}{tp + fn + fp + tn} \tag{2.1}$$

$$precision = \frac{tp}{tp + fp} \tag{2.2}$$

$$recall = \frac{tp}{tp + fn} \tag{2.3}$$

$$F\ measure = \frac{(1 + \beta^2) \cdot precision \cdot recall}{(\beta^2 precision) + recall}. \tag{2.4}$$

$$F_1\ measure = \frac{2 \cdot precision \cdot recall}{precision + recall}. \tag{2.5}$$

The accuracy is often applied by counting the correctly classified tokens compared to all tokens. The precision provides a measure how correct the output of the information extraction model is. The recall states how much information was found. Finally, the F measure provides a score describing the quality of the model.

The evaluation scores of information extraction models reflect often an aggregated value over several scores, especially if the model is responsible to identify more than one type of information. The scores for a specific type of entity need to be aggregated in order to obtain a single value representing the model's quality for the complete task. Typically, two kinds of averages are applied: the micro-average and the macro-average score. The difference between both calculations is the set of values responsible for averaging the values of different types of information. Using a micro-average F_1 measure, the values for true positives, false positives and false negatives are summarized and thus the resulting F_1 score represents an average score for all types of entities, which is weighted concerning the amount of occurrences. In contrast to this, a macro-average F_1 is calculated by averaging the F_1 scores of the different types of entities. Hence, this score does not include a weighting for the actual amount of entities. A differentiation of these averaging scores is only required if several types of entities should be extracted or if some types of entities occur more often than other types of entities, or if some entities generally contain more tokens than other entities. In the domain of reference segmentation, for example, the author contains more tokens than the date. When using a token-level macro-average F_1 measure, the resulting score is biased for high scores concerning the date. Another aggregated score for describing the quality of an information extraction model is the instance accuracy. Here, a true positive is a complete document, in which all entities have been correctly identified. Following this, the instance accuracy provides a measure how many document have been processed without a single error.

2.1.4 Architectures

Information extraction models are often part of a larger pipeline of components. The statistical models or rules build upon annotations added by the previous components that concern most often a linguistic analysis of the document. These components include, for example, tokenizers, sentence detectors, gazetteers, part-of-speech taggers, morphological analyzers or parsers. Information extraction systems are therefore integrated in an architecture or framework for natural language processing in the majority of cases. The architectures provide uniform models for data exchange, component interfaces and process control, which simplify the interoperability of the components and facilitate the specification of complete pipelines. Figure 2.1 depicts a typical pipeline for natural language processing and information extraction. The documents are provided by readers that are able to process different formats. Different phases add additional annotations for linguistic, morphological and syntactic information. In the semantic analysis step, information extraction models like named entity tagger extract interesting information using the annotations provided by the previous phases. Finally, the entities are stored by writers in databases or similar sinks. This section provides a short summary of popular architectures for natural language processing[11].

2.1.4.1 UIMA

The Unstructured Information Management Architecture (UIMA) [77] is a flexible and extensible framework for the analysis and processing of unstructured data and text in particular. It directs its attention especially to the interoperability of components and the scale-out functionality for vast amounts of data. The components of the framework are called analysis engines and are specified in a descriptor that provides information about their implementation, configuration and capabilities. The analysis engines communicate in a pipeline by adding or modifying the meta information stored in the CAS, which contains the currently processed document. This information is represented by typed feature structures and is ordered in indexes for an efficient access. The type of the feature structure defines its semantic and its additional features, which can consist of primitive values or other feature structures. The available set of types and their inheritance is specified in type system descriptors. The most common type of a feature structure is the annotation that defines two additional features, begin and end, assigning its type and additional features to a span of text. Most analysis engines create new annotations or modify existing ones in order to represent the result of their analysis.

UIMA is only a framework and does not ship a rich selection of components. There are, however, component repositories like DKPro [95], ClearTK [95] or uCompare [111] that provide analysis engines of well-known components for natural language processing, e.g., the Stanford CoreNLP Natural Language Processing Toolkit [147], the Mate Toolkit[12] or OpenNLP[13]. The development of Java-based components is facilitated by different extensions like uimaFIT [156]. Two prominent applications that build upon UIMA are the DeepQA system Watson [78] and the clinical Text Analysis and Knowledge Extraction System (cTAKES) [179].

[11] Parts of the description of this section are taken from Peter Kluegl, Martin Toepfer, Philip-Daniel Beck, Georg Fette, Frank Puppe. UIMA Ruta: Rapid Development of Rule-based Information Extraction Applications. Natural Language Engineering, 2014 [125].

[12] `https://code.google.com/p/mate-tools/`

[13] `https://opennlp.apache.org/`

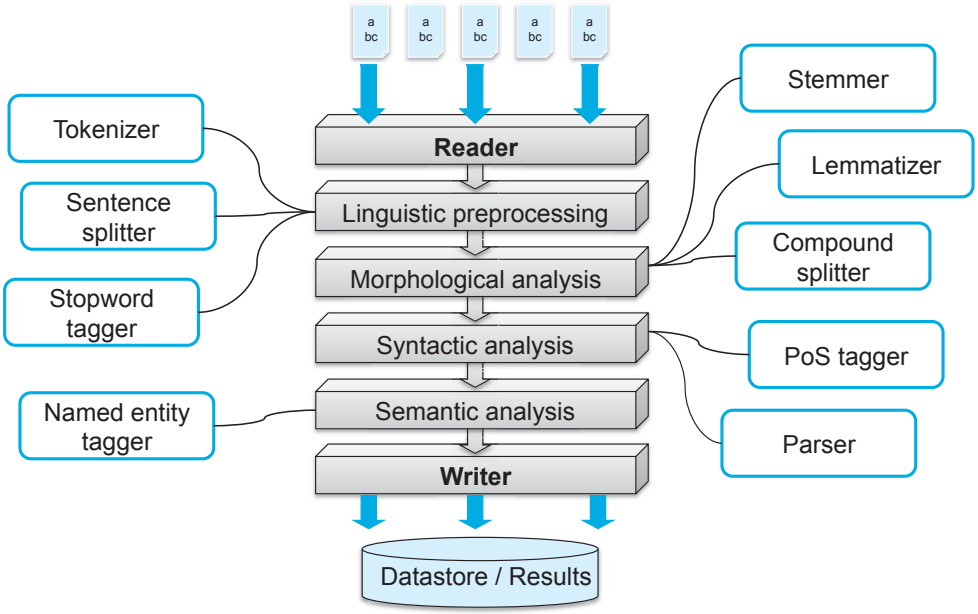


Figure 2.1: Typical pipeline architecture for natural language processing[14].

2.1.4.2 Other Architectures

The probably most popular architecture for natural language processing is the General Architecture for Text Engineering (GATE) [53, 58]. It provides a well-established community and a large amount of tools and integrated components. The information extraction component is called ANNIE [57] and is implemented using finite state transducers. The framework supports the usage of ontologies [31] and provides various functionality for queries [59], information retrieval [56], multimedia processing [67] and machine learning [139].

Other architectures for natural language processing are the flexible and extensible architecture for linguistic annotation called ATLAS [21] and the middleware architecture for the integration of deep and shallow natural language processing components called Heart of Gold [37].

[14] The figure is taken from the tutorial slides of Richard Eckhart de Castilho: `https://dkpro-tutorials.googlecode.com/svn/GSCL2013/tags/latest/slides/GSCL2013UIMATutorialUKP.pdf`

2.2 Rule-based Information Extraction

Components for natural language processing and information extraction nowadays often rely on statistical methods and their models are trained using machine learning techniques. However, components based on manually written rules still play an important role in real world applications and especially in industry [44]. The reasons for this are manifold: the necessity for traceable results, the absence or aggravated creation of labeled data, or unclear specifications favor rule-based approaches. When the specification changes, for example, the complete training data potentially needs to be annotated again. In rule-based components, adaptions in a small selection of rules typically suffice. Rule-based approaches are also used in combination with or when developing statistical components. While models are often trained to solve one specific task in an application, the remaining parts need to be implemented as well. Furthermore, rules can be applied for high-level feature extraction and for semi-automatic creation of labeled datasets. It is often faster to define one rule for a specific pattern than to annotate repeating mentions of a specific type[15].

This section gives an introduction to a selection of prominent systems for rule-based information extraction. First, a detailed description provides an overview of different representatives for rule languages. Development support and other languages are shortly summarized afterwards. As a step towards the next section, algorithms for automatically inducing rules conclude the section.

2.2.1 Rule Languages

Rule-based information extraction systems mostly consist of a specification of a text-based rule language and an interpreter, which is able to apply the rules on documents in order to identify new information [6]. The textual representation of rules leads to a development process where a knowledge engineer manually writes rules. These rules are composed of a condition part and an action part. The condition of the rule is a pattern of properties, which need to be fulfilled by an interesting position in the document. The properties are normally represented as annotations. These annotations assign a specific type and possibly additional features to a span of text. Such features may be capitalization, part of speech, formatting, or presence in a dictionary. Since the sequential order of properties is very important for the specification of patterns, the condition part often represents a regular expression over annotations possibly extended with additional constraints. If the regular expression matches on a text position, then the action part modifies the annotations. In the majority of cases, new annotations are added, which represent interesting entities and relations, or complex properties.

The matching algorithm of a set of rules (rule grammar) is often implemented as a Finite State Transducer (FST), an automaton which traverses the annotation lattices and creates or modifies annotations [54, 68, 26]. The automaton processes the document just once and does not react on its own modifications. A popular strategy is the usage of cascaded rule grammars, where one grammar is based on the results of previous grammars. This approach provides many engineering advantages, for example the easier specification of complex patterns by describing

[15] Parts of the content of this section are taken from Peter Kluegl, Martin Toepfer, Philip-Daniel Beck, Georg Fette, Frank Puppe. UIMA Ruta: Rapid Development of Rule-based Information Extraction Applications. Natural Language Engineering, 2014 [125].

them as a combination of simpler ones, or the clear separation of the stages of engineering approaches and their contexts [27].

2.2.1.1 CPSL

The Common Pattern Specification Language (CPSL) [6] is the result of the effort of different researchers from the TIPSTER research sites for defining a system-independent language for information extraction. At that time, many of the rule-based approaches have been based on finite-state grammars, which represented some sort of regular expressions on the lexical features. These rule-based systems applied, however, different and incompatible formalisms making the translation and reuse of existing systems a difficult and tedious tasks. The CSPL provides a common formalism for the representation of finite state grammars. The separation of the syntax and interpretation should lead to an increased reusability of rule-based systems. While the syntax for defining information extraction rules is accurately declared, the interpretation of the rules is specified indirectly in order to allow arbitrary implementations and adaptions.

The semantics of the rules are determined by an interpreter. The implementation of such an interpreter has to provide certain functionality, which specifies how the rules are applied (cf. [6]):

1. The interpretation of the rules leads to cascaded finite-state transducers.
2. The input of a transducer consists in a sequence of annotations.
3. The transducer verifies if the next annotation in that sequence provides the necessary properties, which have been specified in the rules.
4. The output of a transducer is a sequence of annotations.
5. A cursor points to the currently observed position in the documents. The interpreter tries to match all possible rules at the position of the cursor, but applies only the rule with the "best match". After the rule is applied, the cursor is moved to the next position.
6. Before the rules are applied, the document is tokenized and annotated with the entries of external dictionaries.
7. The interpreter should be extensible by external functions, which implement application-specific functionality.

A CSPL grammar consists of a list of phases, which are sequentially applied on the document. A phase itself is a set of rules, which are compiled into a finite-state transducer. Each phase specifies the types of annotations that are relevant to the rules of the phase. Additionally, a CSPL grammar can include macros and options for the interpreter. Macros are used to modularize recurring parts of the rule definitions. The most important part of a CSPL grammar is the rule section of a phase. A rule is composed of four elements (cf. [6]):

```
Rule: <rule_name>
Priority: <integer>

<rule_pattern_part> -->
<rule_action_part>
```

The name of the rule is given for convenience. As mentioned in the functionality of the interpreter, only the best matching rule is applied. This rule is determined by the size of its match. If two rules match on the same amount of annotations, then the rule with the higher priority is preferred. In case of equal priority, the rule listed earlier is applied.

The pattern part consists of a mandatory body pattern and optional prefix and postfix patterns. When the interpreter tries to apply a rule on the document, it investigates the current position of the cursor by matching the body part at this position. The prefix part of the rule pattern is, therefore, located before the position of the cursor. If all three patterns of the rule have successfully matched, then the cursor is moved to the next position, which is located after the match of the body part.

All three parts are grouped sequences of pattern elements attributed with labels, which are used by the action part in order to refer to the matched annotations. Different kinds of labels are supported: The symbol ":" indicates a reference to the last matched annotation and the symbol "+:" refers to all matched annotations. An exemplary pattern of the CPSL syntax is given in the following [6]:

```
(("douglas"):firstName "appelt")+:wholeName
```

This rule matches on the annotation with the text "douglas" followed by an annotation with the text "appelt", whereas the first label refers only to the first annotations and the second label to both annotations. This example applies an abbreviation for matching on the lemma of a token. A pattern is normally composed of an annotation type followed by one of its attributes and an optional comparison to a value. Patterns can be combined with different operators for alternation ("|"), iteration ("*" and "+") or optionality "?", which are common elements in regular expressions.

The action part of a rule consists of a comma-separated list of actions, which are composed of the specification of an annotations type and optionally an attribute of that type. The targeted element can be extended with an assignment operator and a value in order to modify a value of the attribute. If no attribute is given, then a new annotation of the given type is created, which can be used in further actions. The action specification is preceded by a label in order to specify the spans of the new annotations, which correspond to the spans of the annotations matched in the pattern part. The action part can also include simple conditional expressions over the attributes of the annotations and their values.

2.2.1.2 JAPE

The Java Annotation Patterns Engine (JAPE) [54] provides finite-state transduction over annotations based on regular expression and is probably the most noted implementation of the CPSL specification. JAPE is part of the GATE ecosystem and is an essential component of ANNIE [57], which led to its wide-spread use. Following the CPSL specification, a JAPE grammar consists of a set of phases, which are sequentially executed. A phase itself is again composed of a set of rules that are compiled into one finite-state transducer. Before the actual rules of a JAPE grammar are applied, the system utilizes a tokenizer for adding an initial set of annotations. These annotations cover, for example, tokens and their properties like capitalization. After the tokenizer, the system applies a predefined list of gazetteers and creates additional annotations for their entries.

Cunningham et al. identify several differences between the JAPE implementation and the CPSL specification [54]:

1. The pattern part may not contain prefix or postfix contexts.
2. The pattern part does not allow function calls.
3. The pattern part does not support abbreviations for lemmas.
4. JAPE provides several algorithms for the rule application. One algorithm follows the specification of CPSL and applies only the rule with the best match. An additional algorithm applies all matching rules independently of their priorities or size of the match.
5. Unassigned labels in the pattern part are ignored in the action part.
6. The action part can include arbitrary Java code.
7. The macro syntax differs and macros can be utilized in the pattern and action part.
8. Grammars are compiled into Java code and are stored as serialized objects.

In contrast to the CPSL specification, the actions of JAPE rules allow arbitrary modifications of the annotations and can also remove annotations. This is accomplished by the usage of Java code in the action part. The effects of the actions are not available until all rules of a phase have been applied. If a rule is dependent on the result of another rule, then it needs to be listed in different phases. This restriction can be relaxed with the usage of macros that encapsulate the complete prior rule.

For applying a set of rules aggregated in a phase, the JAPE system first compiles the rules into a deterministic finite-state automaton. Each rule by its own can be interpreted as a single automaton. Joining the set of rules into one automaton results in a non-deterministic finite-state machine (FSM) with one initial state and several final states, one for each rule. The non-deterministic finite-state machine is then converted into a deterministic finite-state machine. Details about this process can be found in Cunningham et al. [54].

The resulting finite-state machine is applied on the document in order to create or modify annotations. The rule execution can be summarized in the pseudo-code algorithm 1. The algorithm interprets the document and its annotations as an annotation graph, where the nodes specify possible positions and the edges represent the annotations. An initial FSM instance is created and added to the list of active instances. The matching process starts with the first node or position and iterates over all nodes. Each active FSM instance is investigated and, in case that it is in a final state, the FSM instance is added to the set of accepting instances. Then, for all possible sets of annotations starting at the current node, a new instance of the FSM (or a new state of the FSM) is created, which consumes the set of annotations and continues the matching process. This procedure is repeated as long as active instances with valid states are available. Then, the actions of either all accepting instances are applied or only of the instance with the best match according to the size and priority of the corresponding rules. The algorithm continues with the next node afterwards.

The actual matching strategy of the rules can be specified at the beginning of the phase by the so-called control style. JAPE provides five different control styles: “appelt”, “brill”, “all”, “first”, and

Algorithm 1 Simplified pseudo-code of the rule matching algorithm in JAPE [54].

```
startNode ← the leftmost node
active FSM instances ← new instance of the FSM, current node set to the leftmost node
while startNode is not the last node do
    while not done do
        for all instance F_i in active FSM instances do
            if F_i is in final state then
                accepting FSM instances ← F_i
            end if
            identify all annotations starting at current node
            identify all sets of annotations that can be used to advance the matching process
            for all sets of annotations do
                active FSM instances ← new instance F_n of the FSM
                let F_n consume the current set of annotations
                protocol existing bindings
                let F_n advance to the rightmost node of the matched annotation
            end for
            discard F_i
        end for
        set done = true, if set of active FSM instances is empty
    end while
    either execute the actions of all accepting instances or only with the "best match"
    startNode ← next node after match
end while
```

"once". The control style "appelt" corresponds to the initial interpreter specification of CPSL [6] and applies on the best-matching rule of the phase. The default control style "brill" applies all matching rules and continues the match on the next position after the current match. Already investigated annotations are, therefore, consumed by the rules of a phase. The control style "all" is very similar to the "brill" style, but continues the matching process on the next possible position, which is allowed to be a part of the previous rule match. JAPE rules with the control style "first" already fire with the first successful match and ignore optional repetitions. The control style "once" finally terminates the matching process of the complete phase if one rule has fired.

The JAPE syntax and semantics is illustrated with an example of Cunningham et al. [54] depicted in Figure 2.2. The example starts with two macros, which will replace their placeholders before the finite-state machine is built. The first macro named "MILLION_BILLION" is a simple pattern with four alternatives and matches on tokens that indicate a quantity. The second macro "AMOUNT_NUMBER" matches on a sequence of numbers separated by commas or periods. The pattern ends with an optional whitespace followed by an optional occurrence of the pattern specified in the first macro. The rule itself is named "Money1" and its pattern part consists of three elements. First, it matches on the pattern specified in the second macro. Then, an optional whitespace may occur followed by a mandatory annotation created by the gazetteers. These three patterns are labeled with "money". The action creates an annotation of the type "Number"

covering the span specified by the label “money” and it additionally assigns the values “money” and “Money1” to the attributes “kind” and “rule”, respectively.

```
Macro: MILLION_BILLION
({Token.string == "m"} |
 {Token.string == "million"} |
 {Token.string == "b"} |
 {Token.string == "billion"}
)
Macro: AMOUNT_NUMBER
({Token.kind == number}
 (({Token.string == ","} |
   {Token.string == "."})
  {Token.kind == number})*
((SpaceToken.kind == space)?
 (MILLION_BILLION)?)
)
Rule: Money1
(
  (AMOUNT_NUMBER)
  (SpaceToken.kind == space)?
  ({Lookup.majorType == currency_unit})
)
:money  -->
  :money.Number = {kind = "money", rule = "Money1"}
```

Figure 2.2: Example of a JAPE rule with two macros for the identification of amounts of money [54]. The rule matches on text fragments like “49.95€”.

Additionally to the standard implementation of the finite-state transducer, GATE also provides an improved implementation called JAPE Plus[16], which includes several optimizations concerning the speed-up of the rule execution. The FSM generated by JAPE Plus grammars are minimized and compiled. Furthermore, an optimized data structure is applied for representing the investigated annotations and the results of evaluated patterns are stored for each annotation. The speed-up of the JAPE Plus implementation has been measured with the grammars for named entity recognition of ANNIE applied on 8,000 web pages. The average execution of JAPE Plus was in average four times faster than JAPE.

2.2.1.3 SProUT - XTDL

SProUT [14, 68, 132] (shallow processing with unification and typed feature structures) combines the ideas of finite-state transducers, typed feature structures and unification-based grammars. The development of the system was motivated by the need for interoperability of different components and a more expressive rule language while retaining good execution efficiency.

[16] http://gate.ac.uk/sale/tao/splitch8.html#chap:jape

The rules are regular expressions over typed feature structures and allow defining coreference constraints between the different elements of the rule. For execution, the rules are transformed into finite-state transducers based on the implementation of FS devices [161].

The main representation of information is based on typed feature structures (TFS) given by the typed description language (TDL) [133]. A TFS is a collection of features that represent specific properties and their values, which can consist of primitive elements, but also of other features structures. The types of the TFSs are normally part of a type hierarchy supporting the inheritance of the feature definition. Using this representation of information allows one to specify more complex relations since the values of the features are able to point to other TFS, e.g., representing coreference information. Grammar 2.1 provides an overview of the elements for representing and specifying TFSs. In short, each TFS can be addressed by its type. The features and their values can be specified in square brackets. Additionally, coreferences are able to point to other elements using a variable.

⟨*type-def*⟩ ::= ⟨*type*⟩ ':=' ⟨*avm*⟩ '.' | ⟨*type*⟩ ':<' ⟨*type*⟩ '.' | ⟨*string*⟩ ':<' ⟨*type*⟩ '.'

⟨*type*⟩ ::= ⟨*identifier*⟩

⟨*avm*⟩ ::= ⟨*term*⟩ ('&' ⟨*term*⟩)*

⟨*term*⟩ ::= ⟨*type*⟩ | ⟨*fterm*⟩ | ⟨*string*⟩ | ⟨*coref*⟩

⟨*fterm*⟩ ::= '[' (⟨*attr-val*⟩ (',' ⟨*attr-val*⟩)*)? ']'

⟨*attr-val*⟩ ::= ⟨*identifier*⟩ ⟨*avm*⟩

⟨*coref*⟩ ::= '#' ⟨*identifier*⟩

Grammar 2.1: Fragment of TDL used by XTDL [68, 133].

The rule representation language of SProUT is called XTDL and uses several concepts of the TDL. XTDL is essentially a rule language for specifying regular expressions over TFSs together with the consequences in case the expression was able to be applied. Matching on symbols and annotations is extended by the unifiability of the coreferences in the TFSs.

The definition of a rule starts with the name followed by the left-hand side (LHS), which contains the regular expression, the conditions of the rule. The right-hand side (RHS) is separated by the LHS using the symbol "->" and provides the postconditions of the rule. The LHS allows one to specify patterns of TFSs using the commonly supported operators such as sequences, repetitions or optionality. The actual conditions are TFS specifications with specific values for theirs features or variables for coreferences. Each TFS is considered to match successfully if it occurs at the specific position and if the coreference constraint can be fulfilled. Additionally, different rules can be executed within the context of the pattern using the "seek" operator. The RHS consist again of TFSs, which will be created and their feature values will be assigned if the pattern in the LHS has successfully matched. A list of functional operators completes the

RHS. These externally defined functions can be applied for including additional conditions or modifications of the present information. An overview of the syntax of XTDL rules is given in Grammar 2.2.

⟨*rule*⟩ ::= ⟨*identifier*⟩ (':>' | ':/') ⟨*regexp*⟩ '->' ⟨*avm*⟩? ⟨*fun-op*⟩? '.'

⟨*regexp*⟩ ::= ⟨*avm*⟩ | '@seek(' ⟨*identifier*⟩ ')' | '(' ⟨*regexp*⟩ ')' | ⟨*regexp*⟩ ⟨*regexp*⟩+
| ⟨*regexp*⟩ ('|' ⟨*regexp*⟩)+ | ⟨*regexp*⟩ ('*' | '+' | '?')
| ⟨*regexp*⟩ '{' int (',' int)? '}'

⟨*fun-op*⟩ ::= ',where' ⟨*coref*⟩ '=' ⟨*fun-app*⟩ (',' ⟨*coref*⟩ '=' ⟨*fun-app*⟩)+

⟨*fun-app*⟩ ::= ⟨*identifier*⟩ '(' ⟨*term*⟩ (',' ⟨*term*⟩)* ')'

Grammar 2.2: Grammar of the rule syntax of XTDL [68].

Figure 2.3 depicts an exemplary XTDL grammar rule, which is able to detect noun phrases. The left-hand side of the rule consists of a regular expression over three TFSs of the type "morph", which provides various information about the morphological properties of the underlying token. The first TFS is optional and matches on a token of the part-of-speech determiner.

The second TFS specifies a match on at least one adjective optionally followed by additional adjectives. The last TFS matches on one or two nouns. Each TFS of the left hand side are connected by a coreference on different inflation properties of the matched tokens. Thus, the sequence of tokens is only successfully matched, if the tokens are annotated with the correct part-of-speech tags and if they have compatible case, number and gender. The right-hand side of the rule creates a new TFS of the type "phrase" and assigns the category of the last identified noun. Furthermore, the result of the unification of the case, number and gender is stored in a feature "agr".

```
np :> morph & [POS Determiner,
               INFL [CASE #c, NUMBER #n, GENDER #g ]] ?
      (morph & [POS Adjective,
               INFL [CASE #c, NUMBER #n, GENDER #g ]]) *
      morph & [POS Noun & #cat,
               INFL [CASE #c, NUMBER #n, GENDER #g ]] {1,2}
   -> phrase & [CAT #cat,
               AGR agr & [CASE #c, NUMBER #n, GENDER #g ]]
```

Figure 2.3: Exemplary XTDL grammar rule for detecting conform noun phrases [68]. The rule matches on text fragments like "das blaue Auto" (German).

2.2.1.4 AFST

Boguraev and Neff [26] presented an annotation-based finite-state transducer (AFST), which is able to navigate in dense annotation lattices. The more components and interesting annotations are required in a specific domain, the more annotations overlapping the same span will introduce challenges for the pattern matching process. AFST implements the finite-state calculus over typed feature structures as, for example, known by SProUT [68]. It extends the horizontal sequential patterns with navigation in the vertical direction, which enables the user to specify more constraints for the relationship of overlapping annotations. The system is completely based on UIMA (cf. Section 2.1.4.1) and thus straightforwardly supports patterns over typed feature structures. Furthermore, AFST interprets the matching process of the finite-state transducer as a linear path through the annotation lattice, which is provided by the indexes and iterators of UIMA.

The syntax of the grammar rules in AFST is introduced with an example [26] in Figure 2.4. The rule starts with its identifier followed by the symbol "=". The AFST syntax provides no separation in an condition and action part of the rule since the effects of the rule are integrated in the sequential pattern. The rule starts with an empty transition followed by the start marker for a new annotation and ends with an empty transition followed by the end marker.

Sequential constraints are explicitly given with the operator ".". The language supports the usual operators for disjunction "|" or repetition "*". Following this, the rule matches on an optional token with the feature "pos" set to "DT" followed by an optional repetition of tokens with the part-of-speech tag "JJ". Then, either a token with the tag "NN" or with the tag "NNS" needs to be present. If the rule has successfully matched, then a new annotation of the type "NP" is created covering the annotations within the span indicated by the markers.

```
np = <E>/[NP .
       Token[pos=~"DT"]|<E> .
         Token[pos=~"JJ"]* .
           Token[pos=~"NN"] | Token[pos=~"NNS"] .
      <E>/]NP ;
```

Figure 2.4: Exemplary AFST grammar rule for detecting noun phrases [26]. The rule matches on text fragments like "the blue car".

The matching process is implemented using a special kind of iterator, which validates the presence of the given constraints in the stream of annotations. This "typeset" iterator is dynamically generated given the specification of the rules and searches for an unambiguous path through the overlapping annotations. If several annotations start at the same offsets, the annotation with the highest priority is selected following the natural order of annotations in the UIMA framework. The exact matching behavior is specified by grammar-wide declarations:

honour The iterator normally only investigates types mentioned in the grammar rules. This declaration specifies a set of types with are introduced in the filter set of the iterator. The annotations of these types are now available in the iterator and prevent, for example, the matching on covered annotations.

boundary This declaration specifies a set of types of annotations that limit the sequential matching. By default, the rules should not match across sentence boundaries.

focus The matching process can be restricted to certain windows corresponding to the annotations of the declared types. The rules are then only applied within the span of the specified annotations.

match This declaration specifies, which match or how many matches are considered to be successful by the iterator. By default, only the longest match will be tracked.

advance This parameter influences the next start position after a match. Either the rule tries to match at the next position after the begin of the match or at the next position after the end of the match.

Additionally to the horizontal matching of sequences of annotations, AFST also provides functionality for vertical matching. Two operators are available that move the iterator not to the next position, but to conceptually smaller or larger annotation. This functionality is illustrated with the example in Figure 2.5.

```
findG = PName[@descend] .
          Title[string=="General"] .
          Name[@descend] .
            First[]|<E> . Middle[]|<E> . Last[string=="Grant"] .
          Name[@ascend] .
        PName[@ascend] ;
```

Figure 2.5: Exemplary AFST rule for detecting generals with the last name "Grant" [26].

The rule matches initially on an annotation of the type PName and then descends into the span of this annotation using the operator "descend". The match of the annotation is only successful, if the following patterns can be applied within this annotation. Therefore, the rule investigates, if the annotation starts with an annotation of the type Title covering the string "General" followed by an annotation of the type Name. Once again, additional patterns for the span of the Name annotation are applied. This annotation has to start with an optional First annotation, followed by an optional Middle annotation and finally by a Last annotation covering the string "Grant". The iterator is then ordered to navigate again back to the previous layer using the operator "ascend". The rule is thus able to find proper names that start with "General" and end with "Grant" [26].

Additionally to these operators that enable the iterator to navigate in a vertical direction, the AFST language also provides a small set of additional predicates. Figure 2.6 provides three examples: The predicate "costarts" is fulfilled if the matched token is located at the beginning of the sentence. The predicate "covers" validates the presence of an annotation of the type "PName" within the span of the matched "Subject" annotation. The last rule matches on annotations of the type "PName" with the same span as an annotation of the type "NP".

```
Token[_costarts=~Sentence[]]
Subject[_covers=~PName[]]
PName[_costarts=~NP[],_coends=~NP[]]
```

Figure 2.6: Three examples of additional predicates for vertical relations in AFST [26].

2.2.1.5 SystemT - AQL

SystemT [43] is a good example for current trends in research about rule languages for information extraction. Its rule language AQL follows a more declarative approach regarding the definition of patterns and provides a syntax similar to SQL. The rules are not transformed into a finite-state transducer but into an operator graph, which allows the selection of an optimized execution plan (cf. Figure 2.7). Breaking up the strict left-to-right evaluation of the patterns, SystemT is able to achieve a much higher runtime performance. The resulting annotators can be integrated into UIMA. An investigation of the formal model underlying AQL can be found in [75].

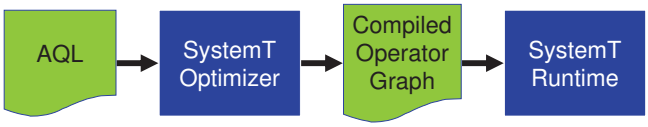


Figure 2.7: Compilation process of SystemT [43]

In contrast to many rule languages that apply grammars to specify the extraction knowledge, SystemT is built on a relational algebra with additional text operators. The data model of the algebra consists of the types span, tuple and relation. A span is a distinguished area of text specified by two offsets. A tuple is defined as a list of spans whereas the size of the list is fixed. A relation consists of a multiset of tuples. The operators of the algebra define atomic creations or consumptions of tuples. Overall, SystemT support 12 operators, which include character-level operators for regular expressions. A complete list of operators is described in Reiss et al [169].

Broadly speaking, AQL rules typically consist of a `create view` statement that specifies the created type of annotation, a `select` and a `from` statement for specifying the input and output, and a `where` statement for the pattern. The rules in Figure 2.8 illustrate the syntax and semantics of the language. The first two statements directly operate on the text of the document. They utilize a regular expression for identifying capitalized words (Caps) and a gazetteer for detecting known last names (Last). The third statement creates new annotations by combining spans of the views Caps and Last. The fourth statement defines the unions of different views in order to merge all annotations for persons in one view. These Person annotations are consolidated by the fifth statement by filtering overlapping annotations using the "ContainedWithin" strategy. Finally, the last statement defines the Person annotations as the output of the query.

```
create view Caps as
extract regexp /[A-Z](\w|-)+/ on D.text as name
  from Document D;

create view Last as
extract dictionary LastGaz on D.text as name
  from Document D;

create view CapsLast as
select  CombineSpans(C.name, L.name) as name
from    Caps C, Last L
where   FollowTok(C.name, L.name, 0 0);
...
create view PersonAll as
  (select R.name from FirstLast R) union all ...
  ... union all (select R.name from CapsLast R);

create view Person as select * from PersonAll R
consolidate on R.name using 'ContainedWithin';

output view Person;
```

Figure 2.8: Excerpt of exemplary AQL rules for the identification of persons [43]. The rules match on text fragments like "Peter Kluegl" if "Kluegl" is listed in dictionary LastGaz.

2.2.1.6 Other Languages

Many other rule languages have been published in the recent years. CAFETIERE [24] combines the strict sequential execution of regular expressions over annotations with basic coreference constraints. Its data model seems to allow only a disjunct partitioning of the document. HIEL [105] is built upon JAPE and focuses on a compact rule representation for elements of an ontology. Xlog [182] is a rule language based on Datalog with embedded extraction predicates. It supports query optimization techniques for an improved runtime performance. The declarative language in PSOX [28] is based on an SQL-like syntax. The system focuses on the extensibility of its operator model, the explainability, and a scoring model for social feedback. Other rule-based systems especially for UIMA are the IBM Content Analytics Studio[17], Zanzibar[18], UIMA-Drools[19] and UIMA Regexp[20].

[17] formerly named IBM LanguageWare Resource Workbench: `http://www.alphaworks.ibm.com/tech/lrw`
[18] `https://code.google.com/p/zanzibar/`
[19] `https://github.com/celi-uim/uima-drools`
[20] `https://sites.google.com/site/uimaregex/`

2.2.2 Development Support

The development of rule-based information extraction applications is an engineering task and should be supported by tooling in order to ensure the efficient specification of rules.

Many of the rule-based systems described in the last section also provide some development tools, which range from simple basic syntax validation to editor support or testing of rules. The most notable tool is probably the development environment of SystemT [42]. It provides an editor with syntax highlighting and hyperlink navigation, an annotation provenance viewer, a contextual clue discoverer, a regular expression learner, and a rule refiner. In WizIE [141], a process model is introduced that guides the developer in the different steps and enables novice developers to create high-quality applications [206].

The Domain Adaptation Toolkit [27] provides grammar development functionality for AFST and is able to create type system descriptors based on the grammars. The development environment for SPRoUT[21] includes functionality for testing.

The IBM Content Analytics Studio is a UIMA-based development environment for the specification of rules in a drag and drop paradigm. ARDAKE[22] provides an environment for the integration of business and semantic rules in UIMA-based annotators. RAD [113] is a tool for Rapid Annotator Development. Its rules are based on inverted index operations [168], which allows for a quick feedback of rule modifications.

2.2.3 Rule Induction

The creation of rule-based information extraction applications is typically a manual engineering process. The rule sets can, however, also be automatically induced. It is overall compelling to automate time consuming and laborious tasks while still being able to investigate the created extraction knowledge. Hence, a large amount of rule learning algorithms have been proposed. These include IEPlus [41], FASTUS [98], AutoSlog [171, 172], CRYSTAL [185], LIEP [102], PALKA [114], WIEN [136], STALKER [155], SRV [82], RAPIER [36], WHISK [186], IEPlus [41], LP2 [45] and BWI [83]. This section gives a short introduction in a selection of these algorithms, which are applicable for different kind of documents: structured, semi-structured and free[23].

2.2.3.1 BWI

BWI (Boosted Wrapper Induction) [83] uses boosting techniques to improve the performance of simple pattern matching single-slot boundary wrappers (*boundary detectors*). Two sets of detectors are learnt: the "fore" and the "aft" detectors. Weighted by their confidences and combined with a slot length histogram derived from the training data they can classify a given pair of boundaries within a document. BWI can be used for structured, semi-structured and free text. The patterns are token-based with special wildcards for more general rules.

[21] `http://sprout.dfki.de/SProUTIDE.html`

[22] `http://www.ardake.com/`

[23] Parts of this section have been published in Peter Kluegl, Martin Atzmueller, Tobias Hermann, and Frank Puppe. A Framework for Semi-Automatic Development of Rule-based Information Extraction Applications. In Proc. LWA 2009, pages 56–59, 2009 [118].

2.2.3.2 CRYSTAL

CRYSTAL [185] learns free-text multi-slot extraction rules named *concept definitions*, which are built of lexical, semantic and syntactic constraints. They operate on sentences or syntactic constituents that are created by a sentence analyzer. Concept definitions are induced in a bottom-up covering manner by generalizing most specific seed rules created from uncovered training instances. A seed rule is merged with the *most similar concept definition* of the initial rule base.

2.2.3.3 LP^2

This method [45] operates on all three kinds of documents. It learns separate rules for the beginning and the end of a single slot. So called *tagging rules* insert boundary SGML tags and additionally induced *correction rules* shift misplaced tags to their correct positions in order to improve precision. The learning strategy is a bottom-up covering algorithm. It starts by creating a specific seed instance with a window of w tokens to the left and right of the target boundary and searches for the best generalization. Other linguistic NLP-features can be used in order to generalize over the flat word sequence.

2.2.3.4 RAPIER

RAPIER [36] induces single slot extraction rules for semi-structured documents. The rules consist of three patterns: a pre-filler, a filler and a post-filler pattern. Each can hold several constraints on tokens and their according POS-tag and semantic information. The algorithm uses a bottom-up compression strategy, starting with a most specific seed rule for each training instance. This initial rule base is compressed by randomly selecting rule pairs and search for the best generalization. Considering two rules, the least general generalization (LGG) of the slot fillers are created and specialized by adding rule items to the pre- and post-filler until the new rules operate well on the training set. The best of the k rules (k-beam search) is added to the rule base and all empirically subsumed rules are removed.

2.2.3.5 SRV

SRV [82] uses single-slot *first order logic* (FOL) rules to classify a given text fragment. It is an ILP system based on FOIL and is suitable for all three kinds of documents dependent on the used feature set. Rules are created in a top-down covering manner from positive and negative instances by starting with a general rule and adding literals until the rule operates well on the training set. It uses a feature-set containing simple attribute-value features and relational features.

2.2.3.6 WHISK

Another multi-slot method is WHISK [186]. It can operate on all three kinds of documents and learns single- or multi-slot rules looking similar to regular expressions. The top-down covering algorithm begins with the most general rule and specializes it by adding single rule terms until the rule causes no errors on the training set. Domain specific classes or linguistic information obtained by a syntactic analyzer can be used as additional features. The exact definition of a

rule term (e.g. a token) and of a problem instance (e.g. a whole document or a single sentence) depends on the operating domain and document type.

2.2.3.7 WIEN

WIEN [136] is the only method listed here that operates on highly structured texts only. It induces so called *wrappers* that anchor the slots by their structured context around them. The HLRT (*head left right tail*) wrapper class for example can determine and extract several multi-slot-templates by first separating the important information block from unimportant head and tail portions and then extracting multiple data rows from table like data structures from the remaining document. Inducing a wrapper is done by solving a CSP for all possible pattern combinations from the training data.

2.3 Machine Learning for Information Extraction

The research in academia on information extraction is mainly dominated by machine learning approaches [44]. The reasons for this fact are manifold. The annotations of labeled examples for supervised machine learning can be performed by auxiliary workers in many domains. In contrast, handcrafted approaches require well-trained engineers. Statistical models are able to include a vast amount of properties and features in the classification decision, which often leads to an improved accuracy. Overall, it is easier to publish approaches using machine learning techniques since no human factor needs to be evaluated in the experimental setup and the experiments can be reproduced given the algorithms and the labeled data.

This section provides a short overview of approaches for information extraction based on supervised machine learning techniques. After an introduction to the essentials and to the different representations of the information extraction tasks for machine learning models, a more detailed description of Conditional Random Fields is provided. These models are applied in Chapter 6 for implementing approaches for exploiting context-specific consistencies. Hence, this section provides the basics and preliminaries for the part of this work, which is based on machine learning techniques.

2.3.1 Essentials of Machine Learning

Information extraction is defined as the identification of well-specified entities and relations. From the perspective of machine learning approaches, this definition corresponds to the classification of text fragments to a given class or label. Hence, machine learning approaches for information extraction typically employ supervised training of a model. The parameters of the models are estimated according to the given annotated training data in order to predict a well-defined set of labels or entities. The process model for supervised machine learning is depicted in Figure 2.9.

The process model can be partitioned into two separate subprocesses (cf. Figure 2.9, part (a) training and part (b) prediction). In the training process (a), a machine learning algorithm is applied in order to estimate the parameters of a model that is later applied for the prediction. The input of this subprocess consists in labeled examples, which typically correspond to annotated documents in information extraction. The input data is transformed into a feature representation

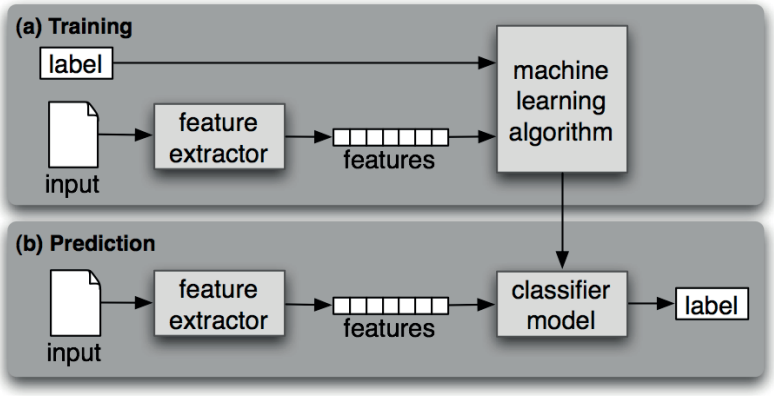


Figure 2.9: Process model of supervised machine learning [22].

using a predefined set of feature extractors. The machine learning algorithms use this feature representation and the additional labels in order to estimate the parameters of the model, which most often includes a weighting of the features for a specific label or class. The output of this process is the classifier model with the learnt parameters. The second subprocess (b) describes the prediction of the labels for new and typically unseen data. First, the same feature extractors transform the unlabeled data into the feature representation, which is known to the model. Then, the model applies inference on the learnt parameters in order to predict the label. In information extraction, the input data consists of documents, and thus the prediction of the classifier model is typically comprised of a more structured output. The classifier assigns a label to each token or sequence of tokens, which potentially leads to variants of multi-instance and multi-label classification.

The success of the machine learning process model for information extraction tasks mainly relies on three factors:

Selection of feature extractors The representation of the documents needs to provide discriminative properties so that the classifier can distinguish different types of entities, positions that contain entities and positions that contain no entities. Popular and commonly applied features for information extraction include the string of the token, its capitalization, regular expressions for specific tokens, ngrams of tokens as well as characters, presence in a dictionary or arbitrary linguistic preprocessing. Additionally, these features are also added for neighboring tokens in a fixed window in order to weaken Markov assumptions.

Selection of model and algorithm The learning algorithm and the applied model need to be able to represent the information extraction task. Different techniques provide different advantages and disadvantages. Discriminative graphical models, for example, are typically preferred over generative models if the classification decision includes many interdependent features. The selection of the model often depends on the specification of the labels in order to represent the information in the textual data. Machine learning techniques

applied for information extraction include Naive Bayes [109, 173], Logistic Regression and Maximum Entropy classifiers [23, 109], Support Vector Machines [49, 180, 96], Random Forests [33], Hidden Markov Models [72, 70, 167, 166], Maximum Entropy Markov Models [151], Conditional Random Fields [137, 190, 201], Markov Logic Networks [170] and Factor Graphs [135, 152].

Labeled data The amount, quality and representativeness of labeled data is an essential factor for the success of supervised machine learning for information extraction. If the given examples are insufficient or do not cover specific aspects, then the trained model will not be able to predict the correct labels. In general, the necessary amount of labeled data increases with the amount of features, number of labels and the complexity of the model.

2.3.2 Representation as a Machine Learning Task

Many different possibilities exist for representing information extraction as a machine learning task. Figure 2.10 depicts the most common and established approaches for applying models on textual data. They can be classified into five categories: classify candidate, sliding window, boundary models, finite state machines and wrapper induction. These approaches are shortly introduced in the following.

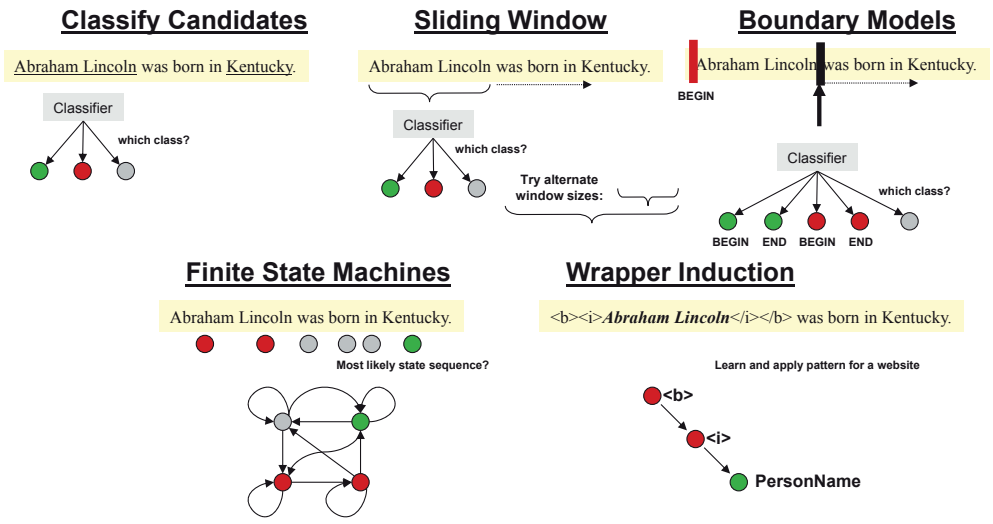


Figure 2.10: Different approaches for identifying information in textual data[24].

2.3.2.1 Classify Candidates

The most straightforward and one of the earliest approaches to represent information extraction as a machine learning task is candidate classification. Here, the identification of information is

[24] Figure taken from the slides of Kamal Nigam: `http://videolectures.net/mlas06_nigam_tie/`

simply defined as a classification task. In general, all kinds of classifiers can be applied, e.g., Naive Bayes [109, 173], Logistic Regression and Maximum Entropy classifiers [23, 109], Support Vector Machines [49, 180, 96] or Random Forests [33]. If several types of entities should be extracted, then either a model is selected that supports this multi-label classification or several classifiers are trained, one for each kind of entity. This approach requires the generation of possible candidates, which contain the information that should be extracted. These candidates may consist of single tokens, complete sentences or chunks identified by parsers. The classification models perform inference on the feature representation of the candidate and possibly of its surrounding text in order to decide if it contains information and which class or label the information possesses.

2.3.2.2 Sliding Window

This approach is very similar to candidate classification. The only difference consists in the generation of the candidates. While the candidates need to be provided by some external component in the first approach, the possible candidates here are generated following a specific procedure. Given a histogram of possible length of entities, a sliding window considers different spans of the documents as candidates. The first candidate consists, for example only of the first token. The second candidate also includes the second token and so on until a given length is exceeded. This procedure is repeated for all tokens of the document resulting in a large amount of possible candidates that cover spans of different sizes.

2.3.2.3 Boundary Models

Similar to the previous approaches, the boundary models also apply classification. However, the problem to process sequential data is solved by classifying the boundaries of the entities. Typically, at least two classification models are trained. One classifier detects the beginning of an entity of a specific type and the other classifier identifies the end of these entities. The actual span and content of an entity is given by the combination of a beginning and the next end. This approach is often improved by adding additional classifiers that identify the end dependent on given beginnings and vice versa. Furthermore, other classifiers that consider the length and the content of a span may be applied in order to avoid unreasonable combinations of boundaries.

2.3.2.4 Finite State Machines

Finite state machines and graphical models are probably the most popular approaches to solve information extraction as an machine learning task. They typically unroll a graph structure over a document whereas one node or random variable corresponds to one token of the text. The estimated label of the variable specifies the kind of entity. It is obvious that this approach provides several advantages, but also some disadvantages. This approach is able to jointly classify different types of entities and thus is able to model dependencies between them. Furthermore, inference over graph structures confirms better with the sequential properties of textual data. On the other hand, these models need to include reasonable encodings of the label set in order to distinguish subsequent entities of the same type. An exemplary encoding is BIO (begin, in, out/other) where two labels are used for each type of entity: one label for the first token of an entity and one label for all other tokens of an entity. Therefore, the spans of entities can be reliably identified. Another problem of these approaches consists in overlapping entities. The graph structures typically

models only one variable for each token. Thus, it is problematic to assign two labels to one token without drastically increasing the complexity of the model. An easy solution for this problem is the usage of separate models for each kind of entity, which, however, prevents the joint inference on different types of entities. A large amount of different models and algorithms exist that can be applied for this approach. These models include Hidden Markov Models [72, 70, 167, 166], Maximum Entropy Markov Models [151], Conditional Random Fields [137, 190, 201], Markov Logic Networks [170] and Factor Graphs [135, 152].

2.3.2.5 Wrapper Induction

Wrappers are typically applied for structured and semi-structured documents like webpages. They specify a combination of attributes or a path in a tree structure in order to classify an entity. The induction of these wrappers often relies on distinct algorithms, rule learners or classifiers. Exemplary methods for this approach can be found in Section 2.2.3.

2.3.3 Conditional Random Fields

Conditional Random Fields are very popular models for solving information extraction tasks. They provide a discriminative approach for classifying sequences of tokens and allow one to include a rich selection of interdependent features. Also more complex graph structures are supported that model long-range dependencies between distant random variables. This section provides a short introduction to different variants, inference, and parameter estimation. The latter two are described in detail only for linear-chain models. Inference and parameter estimation for arbitrary graph structures are shortly outlined with references to further reading. The description summarizes the work of Sutton and McCallum [190, 191][25].

2.3.3.1 Modeling

Conditional Random Fields (CRFs) [137] are undirected graphical models, which model conditional distributions over random variables $\mathbf{y}$ and $\mathbf{x}$. Given exponential potential functions $\Phi(\mathbf{y_c}, \mathbf{x_c}) = \exp\left(\sum_k \lambda_k f_k(\mathbf{y_c}, \mathbf{x_c})\right)$ a CRF assigns

$$p_\theta(\mathbf{y}|\mathbf{x}) = \frac{1}{Z(\mathbf{x})} \prod_{c \in \mathcal{C}} \Phi(\mathbf{y_c}, \mathbf{x_c}) \tag{2.6}$$

to a graph with cliques $\mathcal{C}$ under model parameters $\theta = (\lambda_1, \ldots, \lambda_K) \in \mathbb{R}^K$. The partition function $Z(\mathbf{x}) = \sum_{y'} \prod_{c \in \mathcal{C}} \Phi(\mathbf{y_c}, \mathbf{x_c})$ is a normalization factor to assert $\sum_y p_\theta(\mathbf{y}|\mathbf{x}) = 1$. The feature functions f_k can be real valued in general. However, this work assumes binary feature functions if not mentioned differently.

When CRFs are applied for information extraction tasks, the model is adapted to the properties of sequential data or textual documents. Therefore, the graph structure is normally restricted to a linear chain representing the sequence of labels that are assigned to a sequence of tokens. The

[25] Parts of the content of this section have been published in Peter Kluegl, Martin Toepfer, Florian Lemmerich, Andreas Hotho, and Frank Puppe. Collective information extraction with context-specific consistencies. In Peter A. Flach, Tijl De Bie, and Nello Cristianini, editors, ECML/PKDD (1), volume 7523 of Lecture Notes in Computer Science, pages 728–743. Springer, 2012 [127].

entities of information extraction tasks are identified by sequences of equal labels. If linear-chain CRFs also model long-range dependencies with additional edges between distant labels, then the models are called skip-chain CRFs [189]. Both models are shortly outlined in the following.

Linear-chain CRFs Linear-chain CRFs [137] restrict the underlying graph structures to be linear sequences, typically with a first order Markov assumption. The assignment of y_t given $\mathbf{x}$ and $\mathbf{y} - y_t = (y_t)_{t=1,\ldots,t-1,t+1,\ldots,T}$ is then only dependent on y_{t-1}, y_t, y_{t+1} and $\mathbf{x}$. The components of linear-chain CRFs are referred to with the index L. The probability of a label sequence $\mathbf{y}$ given a token sequence $\mathbf{x}$ is modeled by

$$p_\theta(\mathbf{y}|\mathbf{x}) = \frac{1}{Z(\mathbf{x})} \prod_{t=1}^{T} \Phi_{\mathrm{L}}(y_t, y_{t-1}, \mathbf{x}). \tag{2.7}$$

Φ_{L} is used to describe the factors of the linear-chain edges that link adjacent labels:

$$\Phi_{\mathrm{L}}(y_t, y_{t-1}, \mathbf{x}) = \exp\left\{\sum_k \lambda_{\mathrm{L}k} f_{\mathrm{L}k}(y_t, y_{t-1}, \mathbf{x}, t)\right\}. \tag{2.8}$$

The discriminative impact of the feature functions $f_{\mathrm{L}k}$ is weighted by the parameters $\theta = \theta_{\mathrm{L}} = \{\lambda_{\mathrm{L}k}\}_{k=1}^{K}$. The feature functions can typically be further factorized into indicator functions $p_{\mathrm{L}k}$ and observation functions $q_{\mathrm{L}k}$

$$f_{\mathrm{L}k}(y_t, y_{t-1}, \mathbf{x}, t) = p_{\mathrm{L}k}(y_t, y_{t-1}) \cdot q_{\mathrm{L}k}(\mathbf{x}, t). \tag{2.9}$$

The indicator function $p_{\mathrm{L}k}$ returns 1 for a certain label configuration and the observation function $q_{\mathrm{L}k}$ relies only on the input sequence $\mathbf{x}$. Thus, a feature function, e.g., that indicates capitalized tokens, can be separately weighted for each label transition. Figure 2.11 contains an example of a linear-chain CRF in factor graph representation, which is applied for the reference segmentation task. The label and token sequence are added for a better understanding. Dependencies of the factors to tokens are omitted for simplicity.

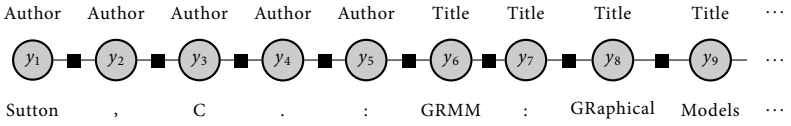


Figure 2.11: A linear-chain CRF applied on the reference segmentation task. The associated labels and tokens are depicted above and below the variables.

Skip-chain CRFs Skip-chain CRFs [189] break the first order Markov assumption of linear-chain CRFs by adding potentials to the graph that address dependencies between distant labels and tokens. A set $I_{\mathrm{x}} = \{(u, v)\} \subset \{1, \ldots, T\} \times \{1, \ldots, T\}$ defines positions u, v for which y_u, y_v are connected by skip edges. The components of skip-chain CRFs are referred to with the index x in order to point out their usage in previous publications, e.g., [189]. The set I_{x} unrolls skip

edges based on token similarity and is therefore only dependent on the token sequence $\mathbf{x}$. In NER tasks, for example, the accuracy can often be increased when the model is encouraged to label similar tokens identically. For controlling the computational cost, I_{x} has to be kept small. An extension of Equation 2.7 with additional skip edges results in the conditional probability

$$p_\theta(\mathbf{y}|\mathbf{x}) = \frac{1}{Z(\mathbf{x})} \prod_{t=1}^{T} \Phi_{\mathrm{L}}(y_t, y_{t-1}, \mathbf{x}) \prod_{(u,v)\in I_{\mathrm{x}}} \Psi_{\mathrm{x}}(y_u, y_v, \mathbf{x}). \tag{2.10}$$

The potentials Ψ_{x} for the skip edges are given by

$$\Psi_{\mathrm{x}}(y_u, y_v, \mathbf{x}) = \exp\left\{\sum_k \lambda_{\mathrm{x}k} f_{\mathrm{x}k}(y_u, y_v, \mathbf{x}, u, v)\right\} \tag{2.11}$$

extending the complete set of parameters $\theta = \theta_{\mathrm{L}} \cup \theta_{\mathrm{x}}$. The feature functions factorize again in an indicator function $p_{\mathrm{x}k}$ and an observation function $q_{\mathrm{x}k}$:

$$f_{\mathrm{x}k}(y_u, y_v, \mathbf{x}, u, v) = p_{\mathrm{x}k}(y_u, y_v, u, v) \cdot q_{\mathrm{x}k}(\mathbf{x}, u, v) \tag{2.12}$$

The observation function enables the model to share observed information between the positions u and v and their neighborhoods, e.g., for providing local evidence at a position where such information is missing. Figure 2.12 contains an example of a skip-chain CRF in factor graph representation, which is applied for the identification of the speaker. The label and token sequence are added for a better understanding. Dependencies of the factors to tokens are omitted for simplicity.

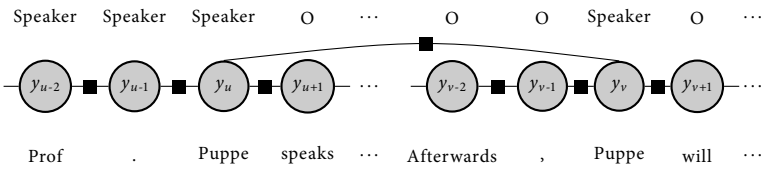


Figure 2.12: A skip-chain CRF applied for the identification of the speaker. The associated labels and tokens are depicted above and below the variables. A long-range dependency connects the random variables y_u and y_v since the corresponding tokens hold the same string "Puppe". The additional factor provides useful features of the context of y_u to y_v, i.e. that the token of y_{u-2} and y_{u+1} are strong indicators for mentions of the speaker.

2.3.3.2 Inference

The inference techniques in probabilistic graphical models need to provide functionality for two tasks. The fist task consists in computing the most likely label assignment $\mathbf{y}^* = \arg\max_{\mathbf{y}} p(\mathbf{y}|\mathbf{x}, \theta)$ for a given input $\mathbf{x}$ and set of parameters θ. This procedure is applied for prediction, when the model is utilized in order to extract information. The second task is the computation of the marginal distribution $p(\mathbf{y}_a|\mathbf{x}, \theta)$ for a subset Y_a of variables. This procedure is required for parameter estimation.

Inference for linear-chain CRFs can be performed efficiently with dynamic programming approaches. Fast inference is essential since it is applied multiple times when estimating the parameters. Furthermore, inference in linear-chain CRFs can be computed exactly and no approximations need to be included in order to retain efficiency.

Linear-chain CRFs mostly rely on the well-known Viterbi algorithm [197] for the two tasks. This algorithm requires the recursive calculation of three types of variables. The forward variables α_t are computed by

$$\alpha_t(j) = \sum_{i \in S} \Phi_t(j, i, x_t)\alpha_{t-1}(i) \tag{2.13}$$

whereas $\alpha_1 = \Phi_1(j, y_0, x_1)$ and y_0 is a fixed initial state. The backward variable β_t is defined accordingly with an initialization $\beta_T(i) = 1$:

$$\beta_t(i) = \sum_{j \in S} \Phi_{t+1}(j, i, x_{t+1})\beta_{t+1}(j) \tag{2.14}$$

The most likely label assignment is computed using the third type of variable δ_t defined as

$$\delta_t(j) = \max_{i \in S} \Phi_t(j, i, x_t)\delta_{t-1}(i). \tag{2.15}$$

Using δ_t, the most likely assignment y^* is calculated by

$$\begin{aligned} y^*_T &= \arg\max_{i \in S} \delta_T(i) \\ y^*_t &= \arg\max_{i \in S} \Phi_t(y^*_{t+1}, i, x_{t+1})\delta_t(i) \quad \text{for } t < T. \end{aligned} \tag{2.16}$$

The partition function is given by $Z(\mathbf{x}) = \beta_0(y_0)$ and $Z(\mathbf{x}) = \sum_{i \in S} \alpha_T(i)$, and the marginal distributions for linear-chain graphs by

$$\begin{aligned} p(y_{t-1}, y_t|\mathbf{x}) &= \frac{1}{Z(\mathbf{x})}\alpha_{t-1}(y_{t-1})\Phi_t(y_t, y_{t-1}, x_t)\beta_t(y_t) \\ p(y_t|\mathbf{x}) &= \frac{1}{Z(\mathbf{x})}\alpha_t(y_t)\beta_t(y_t). \end{aligned} \tag{2.17}$$

When moving to more complex graph structures, inference can become problematic and requires possibly exponential time in worst case. Exact inference is mostly only applied for tree structured graphs. Algorithms for this type of graphs can also be utilized for inference in loopy graph structures, however with the result of potential convergence problems. In general, two different types of algorithms are utilized: Markov Chain Monte Carlo algorithms and belief propagation algorithms.

Gibbs sampling as an example of a Markov Chain Monte Carlo algorithm samples the assignment of one variable while the other variables are fixed. The algorithm can be summarized the following way [191]:

1. Set $\mathbf{y}^{j+1} \leftarrow \mathbf{y}^{j}$
2. For each $s \in V$, resample component Y_s. Sample $\mathbf{y}_s^{j+1}$ from the distribution $p(y_s|\mathbf{y}_{\setminus s}, \mathbf{x})$
3. Return the resulting value of $\mathbf{y}^{j+1}$

This algorithm is initiated with an arbitrary labeling and iterated until a given criteria is fulfilled. A more detailed description of Gibbs sampling can be found in various publications, e.g., [39].

Another important algorithm for inference especially in tree structured graphs, but also in cyclic graphs is belief propagation. This algorithm sends messages between variables and factors that represent a multiplicative contribution to the marginal [191]. Given the subgraph $G_a = (V_a, F_a)$ for every factor index $a \in N(s)$, which contains the variable Y_s, Φ_a and the complete subgraph following Φ_a, the messages m_{as} and the messages m_{sa} are defined as:

$$\begin{aligned} m_{as}(y_s) &= \sum_{\mathbf{y}_{V_a \setminus y_s}} \prod_{\phi_b \in F_a} \Phi_b(\mathbf{y}_b) \\ m_{sa}(y_s) &= \sum_{\mathbf{y}_{V_s}} \prod_{\phi_b \in F_s} \Phi_b(\mathbf{y}_b) \end{aligned} \tag{2.18}$$

The computation of the messages can be performed recursively:

$$\begin{aligned} m_{as}(y_s) &= \sum_{\mathbf{y}_{a \setminus y_s}} \Phi_a(\mathbf{y}_a) \prod_{t \in a \setminus s} m_{ta}(y_t) \\ m_{sa}(y_s) &= \prod_{b \in N(s) \setminus a} m_{bs}(y_s) \end{aligned} \tag{2.19}$$

The marginal distributions for variables and factors are proportional to the product of the received messages [191]. When the graph structure is a linear chain, then belief propagation is equivalent to the well-known forward backward algorithm. A more detailed introduction to the belief propagation algorithm can be found in various publications, e.g., [208].

2.3.3.3 Parameter Estimation

This section gives a short introduction in parameter estimation for linear-chain CRFs. There are several methods like maximum likelihood or stochastic gradients whereas only the former one is considered in the context of this section. Given a labeled training dataset $\{\mathbf{x^{(i)}}, \mathbf{y^{(i)}}\}$, the conditional log likelihood is defined for linear-chain CRFs the following way [191]:

$$\begin{aligned} \mathcal{L}(\theta) &= \sum_{i=1}^{N} \log p(\mathbf{y}^{(i)}|\mathbf{x}^{(i)}, \theta) \\ &= \sum_{i=1}^{N} \sum_{t=1}^{T} \sum_{k=1}^{K} \theta_k f_k(y_t^{(i)}, y_{t-1}^{(i)}, \mathbf{x}_t^{(i)}) - \sum_{i=1}^{N} \log Z(\mathbf{x}^{(i)}) \end{aligned} \tag{2.20}$$

In order to reduce overfitting of the parameters on the training dataset, a penalty term is typically added. These include the Euclidean norm or the L_1 norm. The parameter estimation is performed by maximizing the likelihood. A partial derivative for a linear chain with Euclidean regularizer is [191]:

$$\begin{aligned}\frac{\partial \mathcal{L}}{\partial \theta_k} = &\sum_{i=1}^{N}\sum_{t=1}^{T} f_k(y_t^{(i)}, y_{t-1}^{(i)}, \mathbf{x}_t^{(i)}) \\ &- \sum_{i=1}^{N}\sum_{t=1}^{T}\sum_{y,y'} f_k(y, y', \mathbf{x}_t^{(i)}) p(y, y'|\mathbf{x}^{(i)}) - \frac{\theta_k}{\sigma^2}.\end{aligned} \tag{2.21}$$

The inference techniques described in the last section are applied in order to compute the likelihood and its derivative, i.e., the partition function $Z(\mathbf{x}^{(i)})$ and the marginal distribution $p(y, y'|\mathbf{x}^{(i)})$. Finally, methods like quasi-Newton BFGS [17] are applied to solve the optimization of the concave objective function $\mathcal{L}(\theta)$.

The parameter estimation in arbitrary graph structures is not tractable and thus techniques based on approximations are utilized. Examples for those methods can be found in [191].

Chapter 3

Context-specific Consistencies

Information extraction models often assume that the information in textual documents is independent and identically distributed (iid). Many documents are, however, created in a process that violates these assumptions. Webpages are often generated using templates that specify the arrangement of the textual contents. The entities of a specific type of information are ordered in lists, tables or similar structures and share the same layout. Examples for this are web stores that provide one webpage for each product. The information about the product is arranged with the same layout and formatting in each page. The title of the product may use a different layout than the price, but these entities are consistently structured throughout all pages of a website. Other websites, however, apply different templates for generating the webpages. While the entities in this website are consistently structured, they might apply different layouts compared to the previous website. When an information extraction model processes these websites, for example, in order to extract the product information, then the iid assumption is violated. The entities are not independently distributed since subsets of entities share similar layouts[26].

Such dependencies between entities do not only occur in automatically generated documents. When humans manually compose and write documents, they typically apply some sort of template or consistent structure. They organize repeating entities homogeneously or keep the order of entities. Humans tend to apply the same layout for elements of equal level, like the fonts of different kinds of headlines. These regularities can especially be found in semi-structured documents that include layout in order to highlight different aspects of the text.

The dependencies between entities are called context-specific consistencies in this work. The entities within a context or document share the same or at least similar composition, which manifests in the choice of formatting, the order of entities or other properties. Entities in other documents are also consistently structured, but in a different way. The actual composition of the entities is not known when processing a document. The composition and the applied formatting is potentially contradictory. These consistencies concerning the arrangement and layout of entities are context-specific, because they are only fulfilled in a specific frame or scope. Information extraction models processing these kinds of documents face severe problems resulting in an reduced performance. Properties that indicate a specific type of entity in one document might give evidence for a different type of entity in another document. The discriminative effect of these properties stays thus often behind its potential.

This chapter starts with an investigation of context-specific consistencies and their characteristics. The wide-spread phenomenon of equally structured entities is illustrated by a detailed

[26] Parts of the content and examples have been published in diverse research papers [128, 127, 121, 123].

description of three exemplary domains: reference sections of scientific publications, curricula vitae and clinical discharge letters. These domains are utilized for evaluating the different approaches for exploiting such regularities. Each of the following three sections gives an introduction to one domain by describing the kind of documents in general. Additionally, a closer look at the specific information extraction task, interesting applications and published approaches provides insights in the current research in these domains. Each section also provides a description and examples of the context-specific consistencies with hints how they can help to improve the accuracy of information extraction models. Furthermore, additional domains, in which context-specific consistencies occur, are shortly outlined. The last section of this chapter presents and discusses the related work for exploiting context-specific consistencies, independently of the domain they are applied in.

3.1 Characteristics

A formal definition of context-specific consistencies is problematic because they occur in various manifestations and due to different reasons. A formulation of the problem can be achieved using a generative process like in Blei et al. [25]. However, this model does not capture all aspects of context-specific consistencies. This section investigates the various aspects of these consistencies and provides an indirect approach for formulating the problem.

The main statement of this work can be summarized the following way:

> *Context-specific consistencies cause the consistent composition of entities within a certain context. Entities of different contexts may possess a differing and possibly contradictory composition.*

The statement in second sentence describes the reason why domains with context-specific consistencies cause challenges to information extraction models. The limited validity of features for classifying a certain type of entity prevents that the model exploits its full potential. The elements of the statement in the first sentence are those, which need to be investigated further. Three questions arise: When do context-specific consistencies occur? What is a context? What is a consistent composition of entities? Answers to these questions are given in the remainder of this section.

As the introduction of this chapter already described, context-specific consistencies arise if documents or specific text passages are created in the same process. Possible processes range from applying templates for generating text to authors that manually compose a document. The more structured the texts are, the more likely is the presence of context-specific consistencies. They occur if several entities of one type are arranged in order to increase the readability of the text. Free texts like newswire articles hardly contain context-specific consistencies.

Straightforwardly, the context can be defined as a collection of text that is created by the same process. Since most often the complete document is created by one process, the document provides the context of the entities. It is of course also possible that the context consists of a collection of documents if these have been composed or generated by the same process.

The determination of consistent compositions is more problematic. The overall composition of entities is only similar, but not identical within a certain context. If the entities possess an identical composition in each aspect, then they also contain the same text and all or none of

them are identified. Thus, only specific aspects of their composition are identical. These equal properties need to be described.

A closer look at the properties of entities helps to identify suitable description for their composition. It is often easier to describe parts and specific aspects of the composition. A set of these descriptions is able to model the entire composition of entities. A description from the domain of segmentation of references is the following example: The author begins at the start of the reference and ends directly before the date. This description of the composition of authors consists of two parts. The first part describes the properties of the beginning of an author and the second part concerns the end of the author. No information about the content of the author is given. Hence, the description does not cover all potential aspects, but it is sufficient to specify the consistent composition of authors within one document. The example illustrates two types of descriptions that target a boundary of the entity and the kind of subsequent entities. The following list provides a selection of different types:

- Transition between entities
- Boundaries of entities
- Content of entities
- Order of entities

The transition between entities is the most generic approach for describing the composition of entities. It can also be utilized to model the other types. The concept is similar to the representation of entities in sequence labeling methods, which rate different features for a specific combination of two labels. When the entities are projected on the token sequence of the document, the transition from one token to another can be used to describe specific aspects of the involved entities. This description can be extended with additional properties and features that occur near the two tokens. Tokens of different entities lead to a description of the boundaries of entities and tokens of the same entity represent a statement about the content of the entities. This type of description is well suited for domains where the properties of the composition depend on the neighboring entities. In the domain of segmentation of references, the composition of the entities sometimes depends on combination of the entities. While the pages, for example, begin with a keyword like "pp." in references for publications in proceedings, the entity directly starts with the first page number for articles published in journals. Even if the exact kind of publication is probably unknown when processing a reference section, the combination of entities or labels for tokens can be utilized to achieve the same differentiation. The pages are located after a booktitle entity in references concerning proceedings, but they are following a volume entity in the references to journals. When the transition between entities is applied for describing the composition, it is still possible to provide consistent descriptions since the pages entities are not modelled directly. Instead the transition between booktitle and pages is modelled with the property of a special keyword and the transition between volume and pages is described with a different property, e.g., the occurrence of a number.

The boundaries of the entities can be used as a more robust description in many domains. They model the properties of the first and last token of the entities. Thus, boundaries are a special case of transition between two different kinds of entities. Descriptions using the transition between entities provide some advantages, but there are also disadvantages especially in domains

like curricula vitae. Here, the entities are sometimes located next to each other, but sometimes there are also additional tokens between them. Transition-based descriptions would need to model transitions between an entity and the following entity, but also from an entity to tokens that are not part of an entity. This aggravates the learning process described later due to the reduced amount of examples. It is hardly possible, for example, to induce a valuable and useful description of consistent compositions from one example. Descriptions using only the boundaries of entities do not suffer from this disadvantage. Here, the first and last tokens are described independently of the previous or following entity.

Descriptions for the content of the entities are similar to descriptions based on boundaries. This type is again a special case of the transition-based description since it models the transitions between tokens of the same kind of entity. Descriptions for the content are useful if the consistencies occur for all tokens of an entity. Examples for this consistency are bold or underlined headlines. These properties occur typically also at the beginning or the end of an entity. However, content-based descriptions are potentially more robust especially if all tokens provide consistent features but not the tokens concerning the boundaries. Furthermore, providing information about each token of an entity instead of only the first and last token is able to provide more evidence for graphical models that rely, for example, on a restricted Markov order.

The fourth type of description for the consistent aspects of the entities' composition is the order of the entities. In contrast to the other types, this description does not use properties or features, but only the sequence of entities. In some domains, the entities in a document may not share any similarities that can be specified with the available set of features. However, the order the entities occur may still be consistent within one document. This information can be exploited similar to the other kind of description in order to gain knowledge about the correct appearance of entities.

In summary, the consistent composition of entities can be specified by a combination of different types of description. A subset of the entities' properties or features needs to be identical concerning each type of applied description.

3.2 Domains

Context-specific consistencies are a wide-spread form of dependencies between entities. Due to the common urge to structure information of one type consistently, these dependencies occur in many different domains. This section illustrates this fact by describing three domains in detail. These domains are utilized later in order to evaluate the developed approaches. An outlook to other domains with context-specific consistencies concludes the section.

3.2.1 Reference Sections

Reference sections of scientific publications contain the meta information of the cited papers. This citation metadata or reference string provides specific information about the author, title, publication date and many more entities in order to identify the cited publication. Often the well-known BibTex format is used to define the different fields. The actual occurrence of a type of entity depends on the kind of publication and on the rigor of the editing process. The

reference strings of papers published in proceedings typically contain a book title, whereas papers published in journals specify the name of the journal, the series and the volume.

The references within one reference section follow typically a specific style guide, which determines the intended ordering and layout of entities. The style guide is often prescribed by the conference or journal, to which the paper has been submitted. Different conferences and journals demand different style guides resulting in a huge amount of possibly different composition and layout of the entities. Figure 3.1 contains three examples of different style guides and highlights distinctive aspects of the composition of the references. The reference section in example *A* contains references, which start with the author followed by an italic title. The author and the title are separated by one period, which is, however, an intrinsic part of the author. The date is located near the end of the reference followed by an optional mention of the pages. The reference in example *B* starts with the author followed by the date in parentheses. The title is here also displayed in an italic font, but located after the date. The reference section of example *C* finally specifies author entities that end with a colon followed by the title, which is displayed in a normal font. The date is also given near the end of the reference as in example *A*, but is surrounded by parentheses.

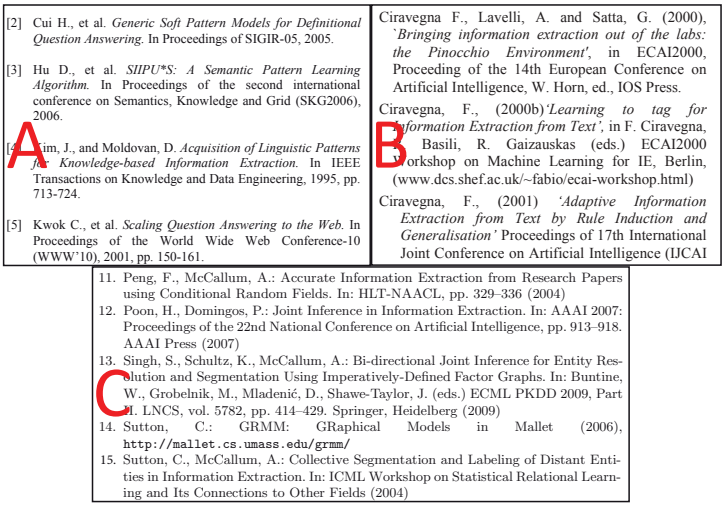


Figure 3.1: Excerpts of three exemplary reference section.

3.2.1.1 Information Extraction Task

There are many interesting and important tasks when processing scientific publications. These include the extraction of the metadata of the paper [153, 93, 87, 142, 210], the identification of the citations in the paragraphs [164], or the disambiguation of the referenced publications [18, 163, 202]. In contrast, information extraction from reference sections consists in the segmentation of the reference string in interesting entities, which often correlate to the BibTeX fields. Commonly, 13 interesting entities can be identified [160]: *Author*, *Booktitle*, *Date*, *Editor*, *Institution*, *Journal*,

Location, *Note*, *Pages*, *Publisher*, *Tech*, *Title* and *Volume*. The specification of entities leads to a complete disjunctive partitioning of the reference string. The set of entities is sometimes reduced or expanded dependent on the requirements of the application. The entities *Author*, *Title* and optionally *Date* are often sufficient for the identification of the referenced paper [199]. By minimizing the amount of different entities, the extraction tasks can be simplified. The remaining entities are then subsumed using an entity like *Venue* (cf. [183]). For a more detailed analysis, more fine-grained entities have to be extracted. One example is the identification of the first name, initials and surname of each author instead of only the complete list of authors (cf. [4]).

3.2.1.2 Applications

The segmentation of references is only one task amongst others that applications need to perform to take advantage of the contained information. First of all, a huge amount of publications is crawled if it is not yet available. Then, the reference section needs to be found within the document and each reference string has to be identified. The reference segmentation task starts here, by labeling the set of references. Finally, the processed references are disambiguated in some applications.

Reference sections of research papers are a valuable source for many interesting applications, which rely on a structured representation of the citation data. The knowledge how often a paper has been cited is a good indicator for its impact in the research community. Scientific search engines like Google Scholar[27], CiteSeerX[28] or Rexa[29] provide useful tools for academic work. They use citation information in a structured representation in order to improve the search results. McCallum et al. have introduced the creation of such an Internet portal in [153]. A considerable amount of research has been spent on creating and analyzing citation graphs, yielding information about research communities and topics (cf. [64, 3, 30, 106]). Furthermore, social bookmarking services like Bibsonomy [101] facilitate the management of bibliographic data and take advantage of the automatic acquisition of citation information. The Icecite Research Paper Management System [10] supports diverse tasks such as metadata extraction, reference segmentation or semantic search.

3.2.1.3 Related work

A vast amount of approaches for segmenting references has been published in academia. The popularity of the task amongst others for evaluating novel information extraction techniques is driven by the availability of labeled datasets and by the importance of the tasks for interesting applications. This section provides an overview on available datasets and related work on segmentation of references by categorizing exemplary publications by their applied technique, focus and domain.

The segmentation of references is a popular task for evaluating information extraction techniques due to the availability of labeled datasets and of the interestingness of the tasks for applications. The ACL Anthology Reference Corpus [20] is a corpus of 10,921 scholarly papers

[27] http://scholar.google.com/
[28] http://citeseerx.ist.psu.edu/
[29] http://rexa.info/

of the area of computational linguistics, which, however, provides no labeled information about the entities. There are several freely available labeled datasets. The CORA Field Extraction dataset [181] contains 500 references from the field of computer science and is annotated with 13 different types of entities. The datasets CiteSeerX [52] and FLUX-CiM [50] are labeled with the same types of entities and consist of 200 and 300 references respectively. The UMass Citation Field Extraction dataset [4] provides references annotated with more fine-grained entities. These cover different elements of the author names and more detailed information about the venue amongst other entities. The references are annotated in a hierarchical manner allowing also coarse-grained access to the entities. The dataset contains overall 1825 references from the fields physics, mathematics, computer science and quantitative biology and thus provides a broader variety of different styles. Kim at al. [115] created a labeled dataset of bibliographic references in digital humanities. The dataset consists of three corpora covering different types of references and citations. Finally, the Plazi dataset [177] contains 25,000 references in more than 1000 publications of the last 200 years.

The CORA Coreference dataset [19] and the CiteSeer dataset [138] have been created for joint inference approaches. They contain 1295 and 1563 references respectively from the field of computer science whereas a not negligible part consists of variants in different representations. The datasets provide additional information about the identity of the references for disambiguation tasks. The CORA Coreference dataset is only annotated with a minimal set of entities since many entities have been merged to *Venue*.

All labeled datasets contain only a list of labeled references and not the complete reference sections. Furthermore, they provide no information from which reference section the references originated. The datasets are, therefore, not applicable for developing or evaluating approaches that try to take advantage of the context-specific consistencies.

The most popular technique are Conditional Random Fields (CRF) [137], which can be considered the state-of-the-art method for this task. Peng and McCallum [160] have been one of the first that applied CRFs for segmenting references and investigated different influence factor of the model. They evaluated their approach on the CORA Field Extraction dataset and achieved an overall token accuracy of 95.37% and a macro-average entity-based F_1 score of 0.915. Councill et al. [52] reproduced these results in their ParsCit system, which is also based on CRFs, and extended the evaluation to two additional datasets. Anzaroot and McCallum [4] and Groza et al. [91] investigated the performance of models trained on different datasets. The former approach is trained on the UMass Citation Field Extraction dataset in order to detect more fine-grained entities. They are able to achieve an overall token-based and entity-based micro F_1 score of 0.978 and 0.912 respectively. Other publications about segmentation of references using CRFs focus on references in different domains [116], layout and formatting [112], acquisition of training data [104], or granularity of underlying tokens [157]. Other works also applied Hidden Markov Models for solving the task [97]. They evaluated their approach on a modified CORA Field Extraction dataset and achieved an micro-average F_1 score of 0.933 based on tokens and an F_1 score of 0.841 based on entities, and a macro-average F_1 score of 0.847 and 0.747 respectively. Ignoring leading and trailing punctuation marks further improved their results.

Early approaches have been based on manually created rule sets for the detection of the entities [86, 65, 108]. However, also later approaches can be categorized as rule-based or knowledge-based. Cortez et al. [50] apply different processes and automatically constructed knowledge bases for solving the segmentation task. This approach is utilized by Afzal et al. [1] for au-

tonomous citation mining. A hierarchical template-based reference segmentation method has been proposed by Day et al. [60]. The approach has been evaluated for six major styles. Chen et al. [40] proposed sequence alignment techniques for segmenting references. Their approach uses the order of punctuation marks and the similarity to citations in a database. Sautter and Böhm [177] introduced an inference mechanism for inducing the applied style using regularities in any list of references. Their experimental evaluation reports an improved accuracy for segmenting references especially for historic reference sections. A more detailed description of the algorithm and its evaluation results can be found in Section 3.4.1.2.

Park et al. published a hybrid two-stage approach for discipline-independent segmentation of references [159]. In the first stage, Support Vector Machines [49] classify the references concerning their style guide. In the second stage, specialized CRFs identify the interesting entities. The necessary training data is acquired with external tools that generate reference representations using different style guides.

The segmentation of references can be combined with other tasks in order to improve the accuracy. Poon et al. [163] and Singh et al. [183] perform the segmentation jointly with their disambiguation. The former approach applied Markov Logic Networks [170], whereas the latter approach is built on imperatively defined factor graphs [183]. There have also been efforts to improve the accuracy using unlabeled data. These models are trained in a semi-supervised fashion incorporating constraints and knowledge about references [16, 15, 146].

Many models for segmenting references have been developed and evaluated using references of scientific publications from the field of computer science. Other approaches cover the task in patent documents [143], web pages [100], digital humanities [116], Chinese electronic books [85], or medical articles in HTML [211].

3.2.1.4 Aspects of Context-specific Consistencies

The references within one reference section typically follow a specific style guide that prescribes the ordering, composition and layout of the entities (cf. Figure 3.1). There is, however, a vast amount of different style guides that define a varying or even contradictory structure of the entities. This conflicting nature of reference sections issues a challenge to information extraction models, which process each reference separately. If the references are segmented collectively, then the consistent composition of the entities within a reference section can help to avoid erroneous extractions. The other references can point out the correct position of an entity in case of doubt. These two assumptions, the challenge for the model and the usefulness of the consistencies, are illustrated with an example.

Conditional random fields are a popular method for segmenting references [52, 91, 104, 116, 157, 160]. The parameters of the models are typically estimated in a supervised fashion using a labeled training dataset. These datasets contain a listing of references with additional information about the location of entities. This information is used to optimize the weights of the features for specific label combinations. The trained model is applied on an instance, which represents a single reference, in order to predict the most likely sequence of labels representing the entities. From the model designers' perspective, the classification process is mainly influenced by the choice of the features. The feature functions need to provide valuable information to discriminate labels for all possible kinds of instances. This works well when the feature functions encode properties that have the same meaning for inference across arbitrary instances. For example in

the domain of reference segmentation, some special words have a strong indicative meaning for a certain task: the word identity feature "WORD=proceedings" always suggests labeling the token as *Booktitle*. Thus, the learning algorithm will fix the corresponding weights to high values, leading the inference procedure into the right direction. Some features, however, violate the assumption of a consistent meaning. The feature that indicates colons, for example, might suggest the end of an author label if current training example finishes author field with colons. However, other style guides define a different structure of the author labels. Consequently, the learning algorithm assigns the weights to average the overall meaning. This yields good generalization given enough training data, but averaging the weights of such features restricts them to stay behind their discriminative potential. Furthermore, ignoring the context of a feature can lead to avoidable labeling errors.

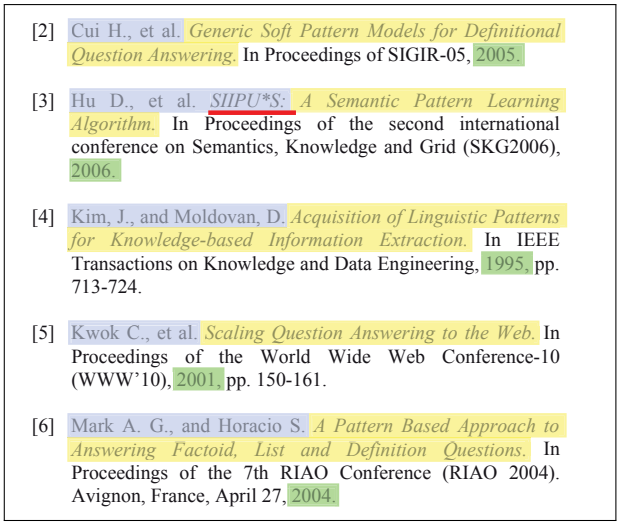
Figure 3.2 depicts an example of some references that have been annotated by a Conditional Random Field with the entities *Author*, *Title* and *Date*. The second reference contains an error since the token "SIIPU*S" is assigned to the *Author* instead of the *Title*. This incorrect labeling was probably caused by the occurrence of the colon near the beginning of the reference in combination of other features. If the complete reference section is taken into account, and if the assumption of consistently structured entities is valid, then the labeling mistake and the correct position of the transition between *Author* and *Title* are obvious. All other references contain an extracted *Author* that ends with a period followed by an italic token. If this knowledge about the consistent composition can be inferred and utilized, then these errors can be avoided or corrected.

This example illustrates only one scenario where the assumption about consistent composition can be helpful. A closer look at arbitrary reference sections reveals that the entities of most types are consistently structured and the order of the entities is identical. The consistencies can also be observed for more fine-grained entities. The similarities in ordering and layout of, for example, first names, initials and surnames can also be exploited. The composition of entities within one reference section is, however, not always completely consistent. Different kinds of cited publications produce references with varying layout and ordering of the entities. The Pages entity contains often varying keywords or punctuation marks if the reference is a journal or a conference publication. Publications like workshop papers that undergo a less rigid editing process contain entities with differing punctuations or even lack some entities resulting in a different ordering of the entities for a specific kind of publication.

3.2.2 Curricula Vitae

A curriculum vitae (CV) or resume summarizes the skills, education, experiences and past employments of a person and is commonly submitted in order to apply for a new employment. Big enterprises and recruiting companies receive a substantial amount of CVs, which need to be processed for further effective usage, e.g., management of the CVs or selection of relevant candidates. An automatic extraction of the relevant information can significantly reduce the manual labor and is therefore an interesting option for many companies.

The extraction of the interesting entities is not a simple task, because submitted CVs use often no predefined format. Online templates for applications are not widely spread. In most cases, a person writes her CV with personal preferences concerning structure and layout. This leads to a rich amount of different formats, in which the interesting information is structured. Different



[2] Cui H., et al. *Generic Soft Pattern Models for Definitional Question Answering.* In Proceedings of SIGIR-05, 2005.

[3] Hu D., et al. *SIIPU*S: A Semantic Pattern Learning Algorithm.* In Proceedings of the second international conference on Semantics, Knowledge and Grid (SKG2006), 2006.

[4] Kim, J., and Moldovan, D. *Acquisition of Linguistic Patterns for Knowledge-based Information Extraction.* In IEEE Transactions on Knowledge and Data Engineering, 1995, pp. 713-724.

[5] Kwok C., et al. *Scaling Question Answering to the Web.* In Proceedings of the World Wide Web Conference-10 (WWW'10), 2001, pp. 150-161.

[6] Mark A. G., and Horacio S. *A Pattern Based Approach to Answering Factoid, List and Definition Questions.* In Proceedings of the 7th RIAO Conference (RIAO 2004). Avignon, France, April 27, 2004.

Figure 3.2: Part of an exemplary references section with highlighted spans for the entities *Author*, *Title* and *Date*. The second reference is erroneous since the token "SIIPU*S" is assigned to *Author* instead of *Title* (indicated by a red bar).

CVs may follow different layouts and structure the contained information in diverse ways, but the information within one CV is typically composed in a consistent layout. This is caused by the fact that humans often format repetitions of equivalent blocks consistently with the same layout. These consistencies can be exploited in order to overcome limitations of the information extraction models.

Figure 3.3 depicts parts of a fictional CV that contains information about the past professional experiences, projects and employments, the interesting information for the task in the domain for this work. The focus is laid on the entities *Company*, *Date* and *Title*. *Company* is the former employer, *Date* specifies the timespan of the employment, and *Title* contains either a short summary or the name of the job. The CV is based on a layout with the *Company* in the first line of an employment block written in capitalized letters. The *Title* follows in the second line with an italic font applied. The *Date* is located on the right given by the corresponding month and year. Other CVs apply, however, a different formatting and arrangement of the entities. This can include amongst others the usage of different patterns for specifying the timespan, contradictory formatting of the entities, or different ordering of entities without clear separations in lines.

3.2.2.1 Information Extraction Task

The information extraction tasks in CVs are dependent on the targeted application and typically cover different kinds of contained entities. These entities range from personal contact information over skills and abilities to eduction and employments. A general specification of entities

WORK EXPERIENCE:

AMERICAN EAGLE — City, State
Sales Associate — present
- Collaborated with the store merchandiser creating displays to attract clientele
- Use my trend awareness to assist customers in their shopping experience
- Thoroughly scan every piece of merchandise for inventory control
- Process shipment to increase my product knowledge

PLANET BEACH — City, State
Spa Consultant — Aug. 2008 - present
- Sell retail and memberships to meet company sales goals
- Build organizational skills by single handedly running all operating procedures
- Communicate with clients to fulfill their wants and needs
- Attend promotional events to market our services
- Handle cash and deposits during opening and closing
- Received employee of the month award twice

HEARTBREAKER — City, State
Sales Associate — May 2008 – Aug. 2008
- Stocked sales floor with fast fashion inventory
- Marked down items allowing me to see unsuccessful merchandise in a retail market
- Offered advice and assistance to each guest

VICTORIA'S SECRET — City, State
Fashion Representative — Jan. 2006 – Feb. 2009
- Applied my leadership skills by assisting in the training of coworkers
- Set up mannequins and displays in order to entice future customers
- Provided superior customer service by helping with consumer decisions
- Took seasonal inventory

Figure 3.3: Excerpt of an exemplary curricula vitae[30].

and information is given by HR-XML[31] and resumeRDF [29].

Information extraction models for CVs are often divided in two stages [209]. While the first stage segments and classifies the different sections of the document, which cover personal information, education or past employments, a second stage performs specialized information extraction within these sections. The information extraction task of this work focuses on the identification of entities in the past employments, especially of the timespan (*Date*) one employee worked for a specific employer (*Company*).

3.2.2.2 Applications

The automatic transformation of the unstructured CVs in a structured representation reduces the manual labor in processing these documents and is able to improve many tasks that are hardly efficient when performed on plain CVs. If the information of CVs is available in a structured form, then it can be utilized for querying, semantic search, and management of

[31] http://www.hr-xml.org/
[31] http://upload.wikimedia.org/wikipedia/commons/c/cc/Resume.pdf

the documents in general. The potential employer, for example, is able to automatically filter uninteresting candidates or to search for candidates fulfilling specific requirements.

The information extraction approach of this work is motivated by an industrial project. The information extraction model has to extract all relevant information in German CVs of the sector of information technology. The entities include the name, contact information, abilities and skills. A special focus is laid on the past employments covering the timespan for which the candidate worked for a specific company or within a sector. This information is linked with the skills and titles related to these projects or employments. The resulting entities facilitate, for example, queries for candidates that have worked in a specific sector in the last 10 years whereas a given set of skills has been applied in these positions. The information extraction process of this application was not completely automated since the identified entities needed to be verified, but the process doubled the efficiency of a complete department.

3.2.2.3 Related work

Information extraction approaches and techniques for transforming CVs into structured representations are not as well-studied as the corresponding research on segmentation of references. Nevertheless, there are publications that address the task due to its considerable relevance in real-world applications.

Ciravegna and Lavelli [46] introduced the LP^2 algorithm for supervised induction of information extraction rules. In one of their case studies that investigates the applicability of the approach in real world applications, they investigated the extraction of contact information in English CVs. The rules have been trained on 250 documents and tested on 50 unseen documents. They were able to achieve a macro-average F_1 score of 0.85.

Kaczmarek et al. [110] describe their project plans for information extraction from Polish CVs. The approach is based on the rule-based system SProUT [68] and targets more different kinds of entities, which cover contact information, employment history, education and skills. Maheshwari et al. [145] extract special skills for an improved performance of resume selection.

An ontology-based approach for English and Turkish CVs has been published by Çelik and Elçi [71]. They apply a pipeline of different phases. The converter transforms the document into plain text. The segmenter identifies the different sections, in which a parser tries to detect the interesting entities. The pipeline is completed by normalization, clustering and classification components. The extraction of entities is mostly driven by matching on the labels of an ontology and by additional rules.

Yu et al. [209] investigate the applicability of different techniques in a cascaded model. Their results indicate that probabilistic sequence models like HMM are suitable for the segmentation of the different sections, whereas classification-based methods like Support Vector Machines perform well for the detection of entities. They evaluated their approach on Chinese documents and are able to achieve an average F_1 score of 0.8 and 0.73 for personal information and educational information respectively. Compared to a flat model, their cascaded model is able to improve the F_1 score by 2.7 and 7.4 percent points.

Kopparapu [130] extracts the six major entities defined in HR-XML, which cover total experience, birthday, passport number, email, skills and qualification. The extracted information is stored in a database in order to improve search. In their management system, the user is able to query documents based on the age, qualification, software skills or experience of the

candidate. The approach is mostly based on dictionary matching in general, sequential patterns for experiences and dates, and classification for names. Their experimental results indicate an F_1 score of 0.78 on a test dataset of 50 CVs.

3.2.2.4 Aspects of Context-specific Consistencies

Humans typically apply the identical layout to repetitive sections or paragraphs. The consistent formatting improves the readability and layout of the complete document. This observation can also be made in CVs. Authors arrange the entities similarly in repetitive sections like the past employments. This consistency of the entities' composition can be exploited in order to avoid deficiencies of the extraction model or to improve its accuracy in general.

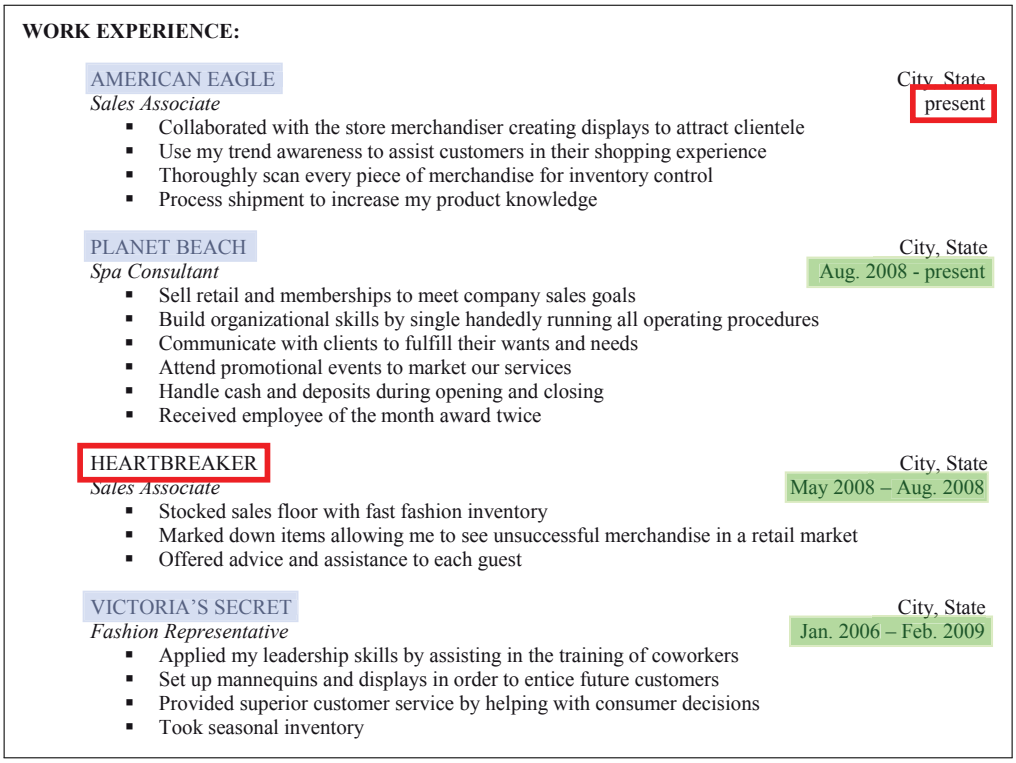
The information extraction models for processing CVs often rely on external dictionaries and word lists. The company, for example, can be identified using simple features, e.g., common suffixes, lists of known organizations or locations. Yet, these word lists cannot be exhaustive, and are often limited for efficiency reasons, e.g., for different countries or only well-known companies. This can reduce the accuracy of the information extraction model, e.g., if the employee had been working for a small company or in another country for some time.

Figure 3.4 depicts the former example of a CV and highlights the results of an inferior model. If the applied dictionaries contain entries for *American Eagle*, *Planet Beach* and *Victoria's Secret*, then these companies can easily be identified without further keywords. However, the company *Heartbreaker* will probably not be extracted since not enough evidence is given due to the missing entry in the dictionary. In addition, the model was able to identify common patterns of timespans but failed to detect the first occurrence "present", which indicated the current position of the author.

In this example, the missing company can simply be identified by analyzing the other companies in this CV. They all contain only capitalized words and are located at the beginning of the section describing one employment. The same applies for the missing timespan of the first employment. All extracted timespans are located in the second line after the title of the job position. By identifying and using the knowledge about the consistent composition of the entities in these sections, many errors of the extraction model can potentially be prevented. Especially precision-driven approaches based on dictionaries and rules can be improved by increasing their recall. However, also probabilistic models are able to take advantage of context-specific hints about the consistent composition of entities for compensating missing features or contradictory formatting.

3.2.3 Clinical Discharge Letters

Clinical discharge letters are created for each patient that leaves hospital. They summarize the diagnoses, history, results of different kind of examinations and observations, therapy, medication and epicrisis. The examinations may cover different tests and measurements like laboratory, ECG, echo or radiography. The clinical discharge letters are therefore a central documentation of the treatment. Furthermore, they are an important source for epidemiologic research and many other important applications like cohort selection, quality management or knowledge discovery. The clinical discharge letters are textual documents in most cases and contain sections for the different results, test and measurements.



WORK EXPERIENCE:

AMERICAN EAGLE — City, State
Sales Associate — present
- Collaborated with the store merchandiser creating displays to attract clientele
- Use my trend awareness to assist customers in their shopping experience
- Thoroughly scan every piece of merchandise for inventory control
- Process shipment to increase my product knowledge

PLANET BEACH — City, State
Spa Consultant — Aug. 2008 - present
- Sell retail and memberships to meet company sales goals
- Build organizational skills by single handedly running all operating procedures
- Communicate with clients to fulfill their wants and needs
- Attend promotional events to market our services
- Handle cash and deposits during opening and closing
- Received employee of the month award twice

HEARTBREAKER — City, State
Sales Associate — May 2008 – Aug. 2008
- Stocked sales floor with fast fashion inventory
- Marked down items allowing me to see unsuccessful merchandise in a retail market
- Offered advice and assistance to each guest

VICTORIA'S SECRET — City, State
Fashion Representative — Jan. 2006 – Feb. 2009
- Applied my leadership skills by assisting in the training of coworkers
- Set up mannequins and displays in order to entice future customers
- Provided superior customer service by helping with consumer decisions
- Took seasonal inventory

Figure 3.4: Excerpt of a fictional curricula vitae with highlighted spans for *Companies* and *Dates*. The boxed areas exemplify missing entities.

This unstructured information has to be transformed in a structured representation in order to be accessible to analytic processes. A first step in this information extraction tasks is the identification of the different kinds of sections present in the discharge letter. The content of the section is typically processed further by additional models in order to extract specific entities or values of measurements. However, the segmentation of the discharge letter and the classification of its sections itself provides already an interesting source of information besides its importance for further processing. It gives insights in the set of performed examinations.

The segmentation of clinical discharge letters can be addressed with many different approaches. One successful approach is the identification of headlines, which can straightforwardly be used to detect the sections. Furthermore, the content of the headlines also provides a good indication about the type of the section, e.g., the kind of examination. The segmentation by headline identification is of course only reasonable in discharge letters that are structured using headlines. If no headlines are used, different techniques like the classification of sentences need to be applied. The clinical discharge letters in the focus of this work, however, provide a headline-based layout in the vast majority of cases. Clinical discharge letters are written by physicians or medical typists, which apply different layout and highlighting in order to represent structure of the letter

and its information. Especially concerning the detection of headlines, layout and formatting can provide essential clues and features for the information extraction model additional to semantic features like keywords. Different authors apply, however, different formatting for the headlines in the discharge letters. Headlines are often displayed in a bold and underlined font potentially in a separate line with a colon at the end. Other documents contain headlines with a different combination of these features or even represent headlines in a completely different and contradictory format. Some authors apply layout features only to emphasize results and not for indicating a headline. The differing formatting of headlines poses a considerable challenge to segmentation models.

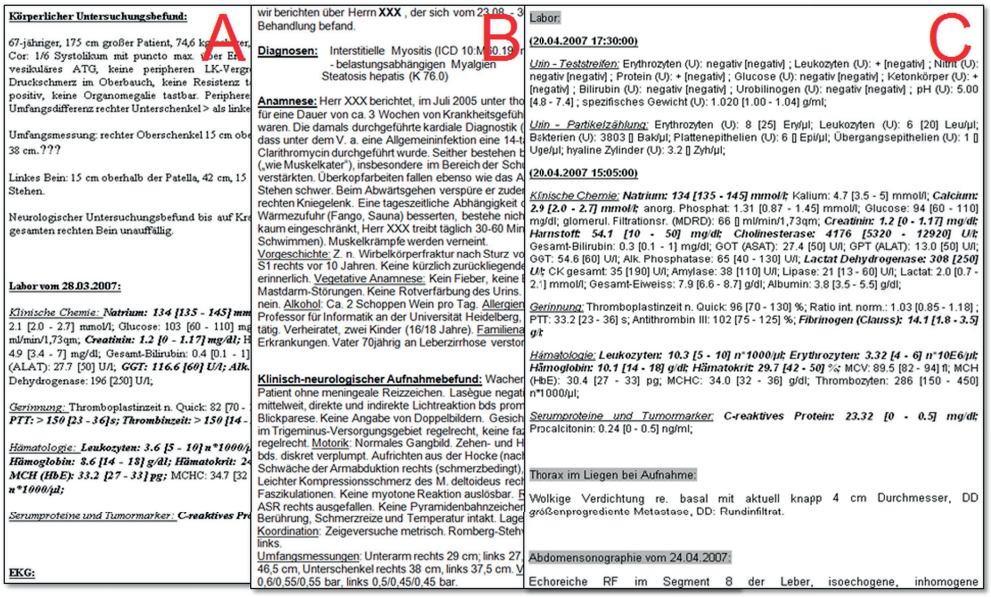


Figure 3.5: Excerpts of three exemplary medical discharge letters from the University Clinic of Würzburg (in German).

Authors of documents with headlines apply typically the same formatting for the headlines of one level. This consistency within one discharge letter can be exploited to improve the segmentation. Figure 3.5 contains excerpts of three clinical discharge letters from the University Clinic of Würzburg. It illustrates the contradictory usage of formatting for headlines across documents and the consistent layout within one document. The first discharge letter (A) applies a classic formatting for headlines. These headlines are bold, underlined, located in a separate line, and end with a colon. The headlines in the second example (B) are also bold, underlined and end with a colon, but the content of the section starts already in the same line. The third discharge letter (C) applies a completely different formatting for headlines. Here, the headlines are highlighted with a different background color. This discharge letter contains also lines with bold and underlined content, which are potentially mistaken for headlines.

3.2.3.1 Information Extraction Task

Information extraction in clinical discharge letters covers many interesting tasks. The extraction is typically performed by a pipeline of diverse components that build on the results of previous processing. The clinical discharge letters are initially anonymized in a de-identification step and segmented in well-defined sections in a segmentation step. These sections are then processed by the following components in order to extract specific kinds of entities. A common specification of the outcome of the information extraction task is given by diverse terminologies or ontologies. Examples for medical terminologies are ICD [88], SNOMED [51, 48] or UMLS [149, 103]. The concepts of these resources may define the semantics of the interesting entities. In many cases, the relations between the entities are investigated. The extracted entities and relations are finally validated and integrated in data warehouses or similar storages.

The task in the focus of this work is the segmentation of the letter by identification of the headlines. This task is a preprocessing step when applying information extraction for the interesting entities in the corresponding section. The segmentation combined with classification of the section in well-known categories of examinations and other observations nevertheless already provides a noteworthy and useful outcome. The interesting entities of this task are straightforwardly the exact spans of the headlines in the document classified according given categories.

3.2.3.2 Applications

The content of clinical discharge letters or any other routine documentation in hospitals is the source of many interesting and important applications. However, the information has to be available in a structured representation for automated analytic processes. As a consequence, information extraction and similar tasks have become of great interest in the medical domain. Evidence for this trend is, for example, the work on freely available pipelines for processing clinical notes [179] or the adaption of the deep question answering system Watson for healthcare applications[32].

The potential applications built on structured information are widespread in the medical domain. The content of the discharge letters can be utilized to select possible candidates for clinical studies. They can be automatically queried dependent on interesting combinations of entities instead of manual investigation, which improves the efficiency of clinical research. Furthermore, the researchers are able to automatically validate their hypotheses using the available data of the patients. They are also able to generate new hypotheses based on previous treatments and examinations. Finally, the structured information can help to improve the quality management of the hospital, e.g., by comparing the diagnoses mentioned in the discharge letters to the actually billed diagnoses.

The information extraction task in the focus of this work, the segmentation of the discharge letters, is a preprocessing step for making these applications possible. The first applications are retrospective cohort studies by querying the data warehouse filled with the entities of the diverse sections of the discharge letters. Segmented and classified sections can also improve many other tasks since they provide context for further processing. An example is word sense disambiguation of specific attributes [192].

[32] `http://www-03.ibm.com/innovation/us/watson/watson_in_healthcare.shtml`

3.2.3.3 Related work

Information extraction in clinical discharge letters and in medical notes in general is a wide and active area of research. A vast amount of approaches and case studies have been published that investigate different tasks in this domain. This section concentrates on related work on the segmentation of the clinical documentation that covers history and physical examinations, discharge summaries and various reports.

The SecTag algorithm developed by Denny et al. [62] is able to identify and categorize explicitly noted sections with headers or headlines and also text segments where these indicators are missing. The algorithm combines different approaches based on rules and naive Bayes that are cascaded in a pipeline. The first phase of the pipeline identifies sentences and listings, which built the input for the following phases. The second phase identifies possible section headers and also their occurrences in text fragments using a predefined terminology for section headers [63]. This process is supplemented by word variants, spelling correction and stop word removal. The third phase applies rules and naive Bayes for scoring the category of the sections. The last two phases apply rules and the Bayesian score for disambiguating and filtering unclear classifications, and for identifying the end of a section. They evaluated the algorithm on 319 "history and physical examination" documents, which contain overall 16,036 sections and 7,858 sections of the predefined categories. The experimental results indicate recall and precision of 0.990 and 0.956 for all sections whereas the values for implicit sections decrease. The exact offsets of the sections have been identified with an accuracy of 94.8 and 85.9% for explicitly mentioned sections and implicit sections, respectively.

Li et al. [140] apply Hidden Markov Models for classifying given sections in outpatient clinical notes. A section corresponds to a variable in the model, which implies a former segmentation of the documents. In an experimental study, they show that the sequential classification that incorporates the order of the possible section categories directly in the model outperforms a similar model that classifies the sections independently of each other. The results indicate an accuracy of 93% for identifying the 15 possible section types in 2,088 clinical notes containing overall 11,706 sections.

Tepper et al. [192] investigate two different approaches based on Maximum Entropy models. The first approach segments and classifies the sections in one step by providing a label encoding for each category. The second approach first identifies a section and then tries to infer the correct category. Both approaches process lines instead of sentences or complete sections, which is grounded in the characteristics of clinical records that are composed of fragments or lists. They have evaluated their approaches on three different datasets for discharge summaries and radiology reports containing 594 to 2,527 sections. They are able to achieve F_1 scores of 0.818 to 0.891 for the complete sections and F_1 scores of 0.878 to 0.929 for headers only. Further experiments indicate that the accuracy of the models significantly decreases if the documents for training and testing origin from different sources.

Mayo clinical Text Analysis and Knowledge Extraction System (cTAKES) [179] also provides a component for the segmentation of clinical notes. The identification of headlines and sections is based on regular expressions and keywords.

3.2.3.4 Aspects of Context-specific Consistencies

The focus of the methods in this work concerning clinical discharge summaries lies on the documents of the University Clinic of Würzburg. In contrast to the documents processed in related work, these summaries typically highlight their sections with headlines and apply rich layout information such as bold fonts. The headlines, which can be utilized for segmentation and classification of the sections, possess a consistent layout as known from semi-structured documents. If the author uses different fonts for specific elements of the summary, then they typically apply always the identical layout to elements of the same category. This includes the headlines of the discharge summaries whereas headlines of different levels may apply different styles. The headlines are, however, only consistently structured within one summary. Different authors structure the summaries with different styles for headlines. This results in a collection of documents with contradictory layouts. This characteristic of layout information with ambiguous discriminative evidence leads to applications that rather process plain text instead of including the layout information in their model. By exploiting the consistent composition of headlines within one document, the segmentation of summaries can be improved.

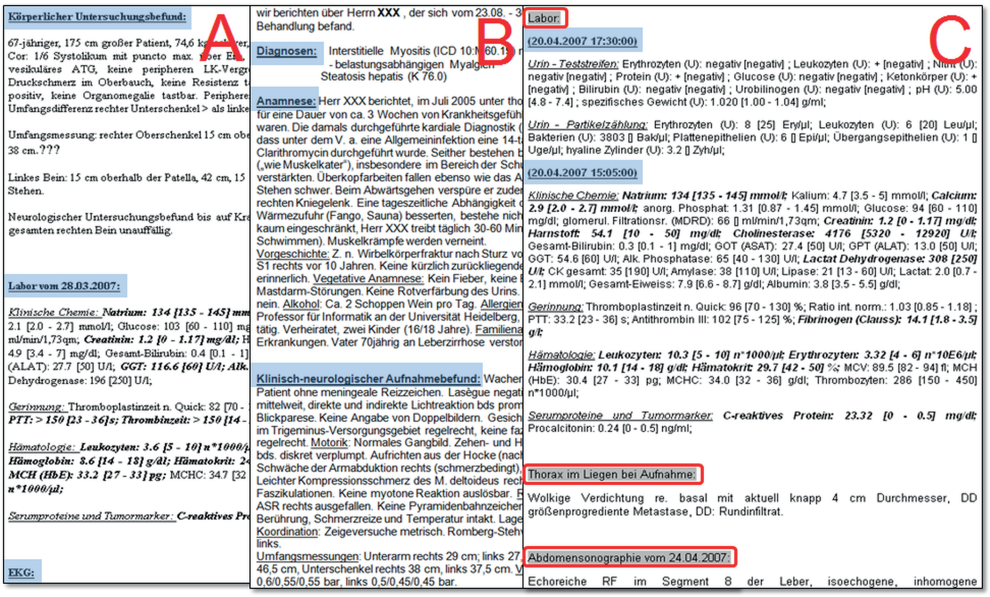


Figure 3.6: Excerpts of three exemplary medical discharge letters (in German) with additional highlighting for correct and missing headlines.

Figure 3.6 depicts the previous examples of discharge summaries with additional highlighting for correct and missing headlines. Example A contains only correct headlines that are bold, underlined, located in an extra line, and end with a colon. The headlines in example B share a similar layout, but are not necessarily located in an extra line. Example C features two false positive headlines that contain information about the date of the laboratory data. These lines

have probably been identified as headlines because of their layout, which covers most properties of the headlines in the other two examples. The actual headlines needed for the segmentation in the third example apply a completely different layout. They use a different background color instead of bold and underlined words. This problem leads to a model, which either overfits on the predominant layout in the dataset or ignores the layout information completely. While the first consequence results in a poor segmentation of summaries with a rare layout, the second consequence causes the model to miss headlines that describe rather infrequent examinations. Both situations should be avoided since they reduce the usefulness of the application.

A closer look at the discharge summaries reveals that most documents contain at least a few headlines describing frequent sections such as diagnoses ("Diagnosen"), history ("Anamnese") or laboratory ("Labor"). These headlines can be reliably identified due to unambiguity of their contained words. A model that identifies these headlines with a high confidence is able to find the remaining headlines by comparing the layout of possible candidates to the layout of confident headlines. If the headline "Labor:" has been identified in example C of Figure 3.6, then the remaining headlines can easily be found by searching for similar layouts. The excerpt contains two more lines that share the same background color and have previously not been extracted. This approach can also improve the segmentation in the first two examples, e.g., for identifying rather unusual examinations or sections with a headline that contains unknown abbreviations or spelling errors.

3.2.4 Other Domains

The last sections provided detailed descriptions of three domains with context-specific consistencies. These domains are only a selection and have been investigated more precisely since they are utilized in this work in order to evaluate the developed approaches. However, context-specific consistencies occur in many other domains. The probably largest accumulation of these consistencies can be found in the web. The webpages of websites are often created based on contents stored in databases. A template or a similar process is applied in order to generate the single pages and to fill them with information. This process straightforwardly leads to dependencies between the contained information. As it was already stressed before, different webpages apply different templates, which leads to contradictory compositions of entities. Examples for these websites are web stores with product information like Amazon[33] or portals for job postings. However, also many other kinds of websites like forums and wikis include context-specific consistencies.

Interesting entities with a similar composition within a certain context can also be found in other types of documents. These include technical documentation, invoices and reports in industry as well as books in general. Compendia that describe different kinds of trees, for example, often structure the information, which tree possesses which attributes like shape of leaves consistently across sections. In the end, the idea of context-specific consistencies can be exploited in order to improve the segmentation of most kinds of semi-structured documents.

[33] `http://www.amazon.com/`

3.3 Exploiting Context-specific Consistencies

The descriptions of the domains with context-specific consistencies in the last section provided already some hints how the consistencies can be exploited in order to improve information extraction for the corresponding tasks. These envisaged solutions and approaches can be abstracted from the specific use case in order to formulate a minimal process model that represents general approaches for exploiting context-specific consistencies. This process model can be summarized with the following three steps, which are performed for each context:

1. Provide an initial prediction of entities.
2. Infer a model for the consistent composition of entities.
3. Consult the model in order to find correct entities.

The necessity of these steps is motivated in reverse order. For improving the extraction of entities under the assumption that they are consistently structured, a model is required that describes the consistent composition of entities and that is able to identify consistent and inconsistent positions. Further on, in order to infer the model, examples of entities are required.

The first step consists in providing an initial set of entities. Two fundamental strategies can be distinguished. Either only highly confident entities are supplied (high precision) or a best possible and realistic prediction of entities is provided (high F score). The differences between these two strategies seem minimal since they both provide positive examples, but they influence the set of applicable techniques in the next steps. The initial prediction needs to fulfill two criteria. It needs to provide a minimal amount of entities so that a model can be inferred at all, and the correctness of the predicted entities may not drop below a certain level so that the inferred model does not provide an incorrect description of the consistencies. The selection of a suitable strategy depends on these two factors. While the first strategy struggles to provide enough entities, the second strategy has to make sure that the prediction contains enough correct entities. The first strategy can be considered to lay its focus on a high precision. Strategies based on a high recall are less useful since they aggravate the induction of a model in the second step. Providing negative examples can also be reasonable, but they are not utilized in this work.

The second step investigates the given examples and learns a description of their consistent composition. Various techniques can be applied for this task. If only one highly confident entity is provided, then specific properties of its composition are selected. For a large amount of entities that also contain incorrect ones, the dominant properties need to be induced from the examples. Here, the capabilities of the learning algorithm to generalize are of major importance.

The third step takes advantage of the induced model. How this can be achieved depends on the techniques applied for information extraction. Rule-based approaches are able to include the information in their extraction knowledge. If only a minimal set of highly confident entities is given, then rules simply annotate additional positions that fulfill the properties stored in the model. Given the second strategy to provide a prediction of entities, rules can utilize the model to process the document anew and hopefully more accurately, or they modify the given entities so that they confirm with the consistencies described in the model. Statistical models are able to take advantage of the consistencies the same way, e.g., by including the knowledge of the model as features or long-range dependencies.

Exploiting context-specific consistencies in general and with this process model in particular is not always reasonable. The benefit of a consistency-aware approach has to exceed its increased development costs. If a context consists of thousands of documents, then it may be more effective to create a separate model for each context and an additional classifier, which is able to select the corresponding model for a given context. The same applies if only few contexts exist. Furthermore, if a context contains only a minimal amount of entities in general, then the advantages of exploiting the consistencies are rather minimal. Either there is no prediction, or if there is a correct prediction, then no entities are left, for which the extraction is improved.

3.4 Related Work

The last sections have given an introduction to three domains, in which context-specific consistencies occur, and described a process model how the consistencies can be exploited. The section concerning the domains provided an overview of related work for information extraction especially in those domains and for a given task. This section considers the related work concerning context-specific consistencies itself, independently of the processed domain or the task. When context-specific consistencies are exploited in order to improve the information extraction performance, the long-range dependencies between distant entities need to be addressed. Collective information extraction also concerns the usage of long-range dependencies in order to improve the accuracy of the models. Thus, these approaches are presented and discussed shortly in order to highlight the differences and the problems when they are applied for exploiting context-specific consistencies.

3.4.1 Context-specific Consistencies

Related approaches that explicitly deal with the challenges of context-specific consistencies are rather sparse and difficult to find. There is no common understanding of the problem statement, which leads to the usage of different terms for describing the ideas of the approach. This work calls the dependencies between the entities context-specific consistencies. Other work refers to scopes, regularities, locales, invariant features, repeated patterns, redundancies, conformance or equal properties. As a consequence, the research on this topic is not well structured and the publications hardly cite each other.

3.4.1.1 Learning with Scope

Blei et al. [25] proposed a hierarchical probabilistic model that captures the ideas of exploiting context-specific consistencies best compared to other publications and also provides a formal process how the documents have been generated. Their approach is motivated by the fact that each website provides structural regularities that differs from other websites. They distinguish two different kinds of features. Global features represent common properties of the text like the current word that provide useful information for the classification task in general. The word "engineer", for example, is an indicator for a job title independently of the website [25]. Local or scope-limited features concern properties that may be applied differently by specific websites and thus are normally not reliable for classification tasks. These features include formatting and layout of the website. While one website represents an interesting entity with a bold font, other

websites might display the same kind of entities only with hyperlinks. Thus, the context is a website with several webpages.

Blei et al. model the information extraction task as a classification problem. They define a graphical model that assigns a label to each token of the document or website, which also provides formatting information additionally to the actual text of the token. This leads to the three lists for word content $\mathbf{w} = \{w_1, \ldots, w_N\}$, formatting $\mathbf{f} = \{f_1, \ldots, f_N\}$ and class labels $\mathbf{c} = \{c_1, \ldots, c_N\}$ for an N word document. The creation of the documents is assumed to be driven by the following process [25]:

1. For each of the K extraction categories:
   a) Generate the formatting feature parameters ϕ_i from $p(\phi_i)$
2. For each of the N words in the document:
   a) Generate the nth class label c_n from $p(c_n)$
   b) Generate the nth word $p(w_n|c_n)$
   c) Generate the nth formatting feature from $p(f_n|c_n, \phi)$

The actual classification of the tokens as class labels or entities is given by the joint distribution over the local parameters, class labels, words and formatting features in Equation 3.1. The additional model parameters are neglected [25].

$$p(\phi, \mathbf{c}, \mathbf{w}, \mathbf{f}) = p(\phi) \prod_{n=1}^{N} p(c_n) p(w_n|c_n) p(f_n|c_n, \phi) \tag{3.1}$$

They provide two approaches for inference since the exact computation of $p(\mathbf{c}|\mathbf{w}, \mathbf{f})$ is infeasible. While the first approach builds upon maximum likelihood estimation, the second approach applies variational methods in order to estimate the conditioned probability. Furthermore, they also provide a discriminative model.

The scoped-learning approach is evaluated for two real-world scenarios. The performance is compared to a baseline naive Bayes classifier using precision-recall curves. The different values of precision and recall are achieved by changing the threshold for the probability, e.g., lowering the threshold leads to an increased recall. The first experimental study investigates the extraction of job titles from 1000 HTML documents, which are split into a testing and training set. A token may include several words and is defined as a set of words with similar formatting. The approach is able to outperform the base line. It achieves a precision of approximately 0.55 at a recall of 0.8 whereas the baseline reaches only a precision of approximately 0.46. The second experimental study concerns a classification task for identifying press releases and is thus not further discussed in this work. It is difficult to estimate the performance of the approach in real-world information extraction scenarios because of the usage of the threshold and the according presentation of the results.

The future work of the publication lists several interesting points. Two of them are especially relevant for the topic of this work. The presented approach does not incorporate the sequence of the tokens and is thus hardly able to provide functionality for common information extraction tasks. Therefore, a sequential model based, for example, on Hidden Markov Models should be

able to achieve superior results. The second improvement concerns the explicit separation of features into two sets. Local and global features are not necessarily disjoint and the affiliation of a feature in a given domain may not be known when specifying both sets. Both items for future work are considered by the contribution of this work. The presented approaches rely on probabilistic models or rules that are able to incorporate the sequential dependencies between tokens and between entities or labels. While the scope-limited features need to be specified for the rule-based approaches, the models based on Conditional Random Fields are able to include all available features. Furthermore, both approaches are able to mix global and local features since they are not restricted due to independence assumptions.

The probabilistic model presented by Blei et al. is hardly applicable for improving information extraction tasks in the targeted domains of references, curricula vitae or clinical discharge letters. First of all, the limitation of generative models concerning the independence of observations prevents the usage of important features in the domains, e.g., combinations of ngrams with formatting and neighboring tokens. Discriminative models like Conditional Random Fields do not suffer from this restriction. They have successfully been applied in many domains and achieved results superior to naive Bayes classifiers or Hidden Markov Models (cf. reference segmentation, Section 3.2.1.3). While this limitation does not prevent the usage of Blei's approach in the targeted domains, it will most likely not be able to compete with state-of-the-art techniques applied in those domains. There are, however, more important reasons why the scoped learning approach is not able to solve the described tasks and to exploit the context-specific consistencies. It is essentially a classification model that requires suitable candidates in order to identify the entities. In the domains of references and curricula vitae, the model would need to classify the tokens individually, which ignores the important sequential dependencies. The regularities and consistencies in the targeted domains cannot be represented by one feature alone. This fact is highlighted by the examples in Section 3.2.1.4, Section 3.2.2.4 and Section 3.2.3.4. The composition of the entities mostly relies on a combination of features whereas also properties of neighboring tokens need to be included. This functionality can only be supported by the scoped learning model if the cross product of the features is used or if more than one observed variable is modelled. Following this, more problems concerning the disjoint sets of global and local features arise. Another challenge that can hardly be solved by the scoped learning model consists in the actual consistent aspects of the entities' composition. Not all tokens of the entities share similar characteristics. In the previous examples, often only the boundaries of the entities can be applied for modeling the context-specific consistencies while all other tokens of the entities do not provide relevant local features. The scoped learning framework can be successfully applied for the segmentation of clinical discharge letters if suitable candidates for the headlines are generated. This domain is, however, already satisfactorily solved by the rule-based approaches of this work and hardly any improvements can be achieved with more complex models.

The scoped learning approach is nevertheless an important and interesting possibility for the future work of approaches in this work. Effort has already been spent on developing a higher-order graphical model that includes the ideas of the scoped learning framework while extending it for representing sequential data. These models have, however, not been able to compete with the machine learning approaches of this work until now.

3.4.1.2 The RefParse Algorithm

Sautter and Böhm [177, 178] proposed the RefParse algorithm for the reference segmentation task that depends less on the emerging style and is thus able to provide improved parsing result for references in historical publications. One central idea of the algorithm consists in the fact that the references of one reference section follow the same style guide. By processing all references of the section at once, the algorithm is able to resolve ambiguities, which results in an improved accuracy. The main functionality of the algorithm relies on patterns and regular expression for morphologically distinctive entities, and on majority votes for inducing the specific aspects of the style guide. Sautter and Böhm call the regularities in reference sections repeated patterns while this work uses the term context-specific consistencies.

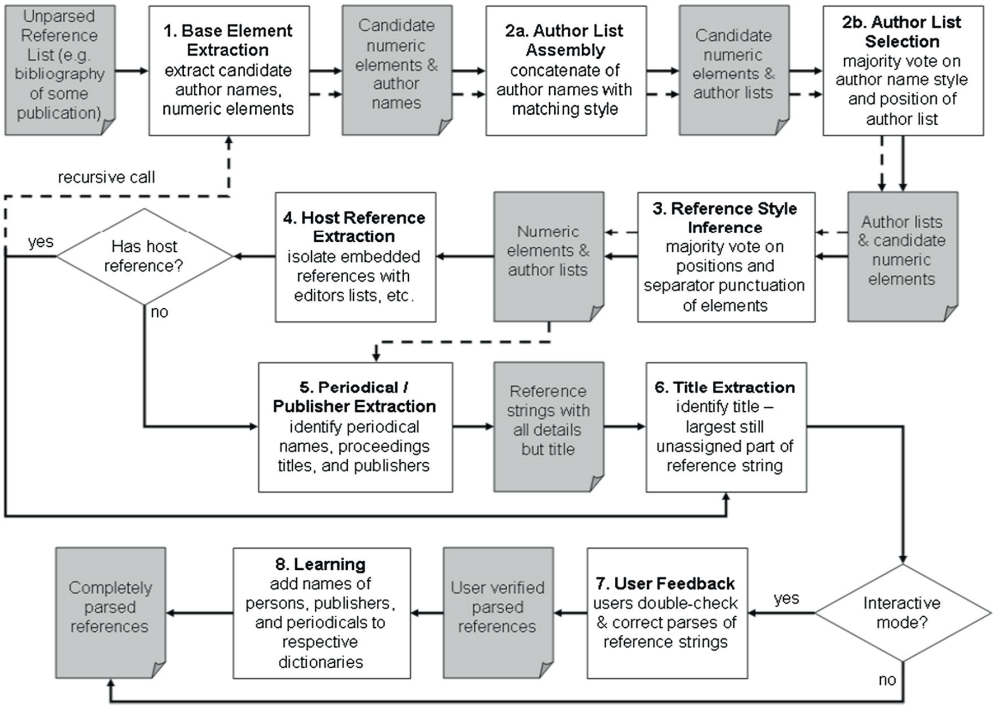


Figure 3.7: The six steps of the RefParse algorithmwith two additional steps for an interactive mode [178]

The algorithm consists of six steps and is summarized in Figure 3.7. The first step uses patterns in order to identify all possible candidates for specific entities, which are reliably identifiable by their morphological structure. These include years of the publication, part designators, pagination and author names. The authors are, for example, detected using different patterns concerning the positioning of initials, first names and last names. Ambiguous and false positive entities are not problematic at this point since later steps will retain only the most likely entity. The second step is divided into two stages. The assembly stage generates all possible author

lists using only names with equal ordering of the elements. The selection stage uses a majority vote over the occurring styles in order to identify the most likely author list. The third step induces the most likely arrangement of the found candidates of entities. The step is again divided into an assembly stage and a selection stage whereas two criteria are applied for the selection. Possible arrangements should contain as many entities as possible and the arrangements should be valid for many references. The fourth step detects embedded references that are used to describe proceedings or books, in which the paper was published. Embedded references are recursively processed by starting again at the first step. The last two steps are finally responsible for identifying periodicals, publishers and titles.

Sautter and Böhm evaluated their algorithm on two datasets. The results on the Cora dataset indicate that the algorithm is able to outperform freely available parsing services for most types of entities. Only the accuracy of the title is decreased. Overall, a word-level accuracy of 91.5% was achieved compared to 83.8% of similar systems. The second dataset Plazi consists of more references in general and especially also of historical references. Here, the algorithm provides superior results. It is able to achieve a word-level accuracy of 94.3% compared to 79.7% of similar systems.

The approach of Sautter and Böhm is similar to the stacked transformation presented in Section 5.2. Both approaches are inherently domain-specific and cannot be applied in other domains or tasks without considerable effort. The RefParse algorithm as well as the rules for the stacked transformations apply majority vote for inducing specific aspects of the context-specific consistencies. While the RefParse algorithm votes on all candidates for single entities or their assemblies, the rules utilize only the entities given by the base component, which are already unambiguous. The inclusion of different candidates potentially leads to more robust parsing of the references. On the other hand, the stacked transformation approach is able to modify just the inconsistent aspects of the entities. The entities are corrected in order to be consistent in contrast to a selection of consistent entities from a given list of candidates. Transformations are able to create the correct entities even without a correct candidate.

The experimental results of both approaches are hardly comparable. The RefParse algorithm was created to extract different types of entities, whereas the author, for example, refers to single authors instead of the author list and is thus more fine-grained. The stacked transformations approach only extracts the four entities author (list), title, editor and date/year. This restriction is not only an advantage since other entities are able to help identifying the targeted ones in the domain of reference segmentation. The results given in Sautter and Böhm [177] use the word-level accuracy while the stacked transformations apply the word-level F_1 measure. Furthermore, both approaches utilize different strategies for the inclusion of punctuation marks and the stacked approach is evaluated on a rearranged dataset that contains a subset of the Cora dataset. Despite all differences in the experimental setting, Figure 3.1 provides an overview of the results of both approaches for convenience.

3.4.1.3 Properties-based Collective Inference

Gupta et al. [94] proposed a properties-based collective inference framework. Their work mainly focuses on Markov Random Fields with symmetric clique potentials. Their additional inclusion of properties provides a more general framework for collective information extraction than similar work (cf. [189, 81, 134, 131, 35]). They assume that each document or webpage in a domain

	RefParse (Accuracy)	Base (F_1 score)	AIE (F_1 score)
Author	98.6%	0.984	0.993
Title	79.0%	0.962	0.979
Editor	74.6%	0.949	0.949
Date	98.8%	0.973	0.983

Table 3.1: Evaluation results of the RefParse algorithm, the base component (of the stacked approach) and the stacked component (AIE) in the Cora dataset.

provides some sort of consistency in the composition of entities of one type. This consistency is modelled with additional properties that are uni-valued within a domain. This means that a specific property has always the same value for one type of entity, but the actual value of the property differs between different contexts or documents. The properties are utilized to specify long-range dependencies in the graphical model.

Gupta et al. evaluate their properties-based collective inference framework in the reference segmentation task with a special focus on domain adaption. The dataset consists of 433 references from webpages of 31 authors. The composition of entities from different websites can greatly differ since different authors apply different style guides and layouts to visualize the references. In addition to the standard features used in this domain, they included four Markovian properties [94]:

1. $g_1(\mathbf{x},\mathbf{y})$ = First non-Other label in $\mathbf{y}$
2. $g_2(\mathbf{x},\mathbf{y})$ = Token before the Title segment in $\mathbf{y}$
3. $g_3(\mathbf{x},\mathbf{y})$ = First non-Other label after Title in $\mathbf{y}$
4. $g_4(\mathbf{x},\mathbf{y})$ = First non-Other label after Venue in $\mathbf{y}$

The properties-based framework is compared to a common baseline sequential model. The results are depicted in Table 3.2 for different allocations of dataset concerning test and training sets. The framework is able to achieve a significant error reduction up to 25% compared to the baseline. Figure 3.8 provides an overview of the combined F_1 error of the labels author, title and venue for the individual webpages (domains). The properties-based framework produces fewer errors in most domains. Only in two domains, the results deteriorate compared to the baseline.

There are two major differences between the properties-based framework and the approaches of this work: the inference technique and the expressiveness of the properties. The model of Gupta et al. apply more sophisticated inference techniques that open up many interesting possibilities for improved models. These include the incremental unfolding of the graphical structure, which allows one to define long-range dependencies dependent on the label sequence. The machine learning approaches of this work utilizes freely available default implementations for the inference in the graphical models. As a result, exploiting intermediate predictions of the label sequence caused convergence problems of the inference. Thus, a more straightforward technique was applied by providing a static prediction of an additional model. On the other

Train	Title		Venue		Author	
(%)	Base	CI	Base	CI	Base	CI
5	0.707	0.748	0.585	0.625	0.741	0.743
10	0.780	0.821	0.692	0.722	0.756	0.759
20	0.858	0.886	0.767	0.789	0.807	0.807
30	0.917	0.930	0.815	0.826	0.877	0.880
50	0.923	0.942	0.835	0.845	0.894	0.900

Table 3.2: Evaluation results of the properties-based framework (CI) and the baseline for different sizes of training sets [94].

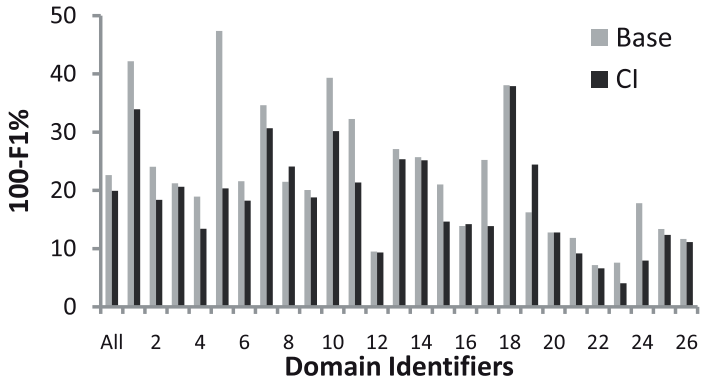


Figure 3.8: F_1 error of the properties-based framework (CI) compared to the baseline [94].

hand, different evaluations highlighted the power of unigram factors instead of skip-edges (cf. Comb-chain CRF, Section 6.3). These simpler models of linear graphs are able to compete with more complex graphical structures with long-range dependencies even if the inference technique was able to converge.

The approaches of this work include more expressive properties in order to exploit the context-specific consistencies. Atomic properties as they are used in the properties-based framework are often not sufficient to describe the consistencies in many domains. The restriction to equal values of the entities properties limits the usage of arbitrary features. The properties need to refer to distinct features or situations like the word content or neighboring labels. The rule-based approaches of this work are able to integrate any feature for representing properties including formatting since the rule sets are already inherently domain-specific. An extension of the properties-based framework leads to an instantiation of specific features to boolean properties, which contradicts the basic ideas of this framework concerning properties independent of their feature-based values. Nevertheless, the representation of the consistencies in the rule-based approaches and the properties-based framework are rather similar. Two entities are consistent if one specific feature is equal near their beginning or end. The machine learning approaches of this work apply a more expressive and generic solutions for modeling the consistent properties.

While the properties-based framework requires specifying the set of properties when the set of features are defined, this work uses template-based descriptions of possible aspects of the consistencies. This strategy can be compared to including various properties for all types of entities. This, however, can quickly cause additional errors when processing documents with incorrect consistency assumptions, e.g., enforcing a property on an invalid aspect of consistency. This problem is solved by allowing more expressive properties that consist of conjunctions of arbitrary features. Furthermore, atomic values or descriptions are not sufficient in many domains in order to specify the consistencies. Often a sequence of word content combined with additional features is required (cf. Section 3.2.1.4, Section 3.2.2.4 and Section 3.2.3.4). Atomic values as they are applied in the properties-based framework and the rule-based approaches are able to improve the accuracy in many situations. However, they deteriorate the performance if the consistency cannot be expressed by a single feature. This result has been recognized when evaluating models similar to the properties-based framework. A Conditional Random Field with additional long-range dependencies based on specific word content or even on weighted similarity of the feature vectors was not able to achieve the results of the other approaches that relied on classifiers in order to induce the properties. The usage of a learning model for identifying consistent and inconsistent positions has another major advantage. It is not only able to validate the consistency of entities, but can also highlight consistent positions without entities with high accuracy. This functionality has the potential to greatly increase the performance of the information extraction model, especially in domains with low recall.

Gupta et al. highlight that their approach is inference-only and does not require retraining the model for domain adaptation. This statement is also true for the machine learning approaches of this work. In the future work, they mention the automated induction of properties especially for handling unlabeled domains. This work learns a model of consistency during inference. Overall, both approaches, the properties-based framework and the machine learning approaches of this work, would benefit from including ideas and techniques of the other one. Combining the intermediate prediction of the label sequence and the robust inference techniques with the more sophisticated modeling of consistencies has the potential further improve the performance of information extraction models.

3.4.1.4 Exploiting Content Redundancy

Gulhane et al. [92] presented an approach for exploiting content redundancy for web information extraction. They gather seed entities from some initial sites and utilize them together with a similarity metric in order to identify new entities. Two properties have been observed by the authors:

> "***Property 1:*** *Multiple sites contain pages for the same entity. Furthermore, the value of an attribute across the various pages for an entity are "textually" similar.*
> ***Property 2:*** *Pages within a web site have similar structure conforming to a common template.*"[92]

They exploit these properties with an Apriori-style algorithm [2] to filter inconsistent entities amongst other things. The experimental evaluations for web information extraction concerning restaurants and bibliographic entries indicate a precision greater than 95% and a recall greater than 80%.

In contrast to the approaches of this work, the algorithm of Gulhane et al. mainly relies on similarity metrics. The general approach can be compared to the increase of recall in precision-driven prototyping (cf. Section 5.1). Some initial entities are utilized in order to identify more entities. The approaches of this work, however, are limited to the currently processed document since exploiting content redundancy hardly improves information extraction in the targeted domains.

3.4.1.5 Other publications

There are several publications that seem to cover the same or similar problems, but they actually utilize only similar terms to describe their problem statement or take advantage of the consistencies for different tasks. Wong and Lam [204] adapt web information extraction knowledge by mining site-invariant and site-dependent features. They utilize the site-invariant features for acquiring training examples of the unseen target websites. Then, both sets of features are applied in learning extraction knowledge for the targeted website. An approach similar to Gulhane et al. [92] was published by Machanavajjhala et al. [144]. They try to process heterogeneous web lists by exploiting content redundancy and layout homogeneity. Rush et al. [175] propose the usage of inter-sentence consistency constraints for improving parsing and POS tagging. They exploit the similarity between test set sentences by including global constraints that reinforce the linkage between surface-level contexts and syntactic behavior. Another approach that exploits test set regularities was published by Slattery and Mitchell [184]. They improve classification in relational domains, like website categorization. Yang et al. [207] incorporate site-level knowledge to extract structured data from web forums. Arnold and Cohen [7] proposed intra-document structural frequency features for semi-supervised domain adaption. While the title indicates the usage of similar ideas, their approach rather considers different parts of the documents and the frequency of the contained words.

3.4.2 Collective Information Extraction

The approaches of this work try to exploit long-range dependencies between entities in a document or context under the assumption that these entities share a similar composition. In order to include the properties of an entity in the extraction of another entity, all related entities need to be processes collectively. Thus, the presented approaches can be categorized as collection information extraction while the actual problem that needs to be solved differs from the related work in this research area[34].

Models of collective approaches for information extraction are often motivated by two assumptions: The labeling of similar tokens is quite consistent within a given context or document since those mentions mostly refer to the same type of entity. The discriminative features to detect the entities are sparsely distributed over the document. Thus, the accuracy to detect different mentions of an entity can be improved by leveraging and transferring their local context to distant positions. However, both assumptions provide no advantages when facing context-specific

[34] Parts of the content of this Section have been published in Peter Kluegl, Martin Toepfer, Florian Lemmerich, Andreas Hotho, and Frank Puppe. Collective information extraction with context-specific consistencies. In Peter A. Flach, Tijl De Bie, and Nello Cristianini, editors, ECML/PKDD (1), volume 7523 of Lecture Notes in Computer Science, pages 728–743. Springer, 2012.[127]

consistencies. Similar tokens are either rather sparse or pose minor problems in the targeted domains. It is more interesting and helpful to investigate the long-range dependencies between entities that contain no similar tokens, but are consistently structured. As for the second assumption about the local context, the entities in the targeted domains share a similar composition and thus provide no new contextual information for the other entities. The approaches of this work rather try to identify the shared aspects of the entities' composition in order to improve the overall accuracy.

Bunescu et al.[35] use Relational Markov Networks and model dependencies between distant entities. They apply special templates in order to assign equal labels if the text of the tokens is identical. The skip-chain approach introduced by Sutton and McCallum [189] extends linear-chain CRFs with additional factors for long-range dependencies. They link the labels of similar tokens and provide feature functions that combine evidence of both positions by which missing context can be transferred. Finkel et al. [81] criticize the usage of believe propagation and apply Gibbs sampling for enforcing label consistency and extraction template consistency constraints. All of these approaches with higher-order structures fight the exponential increase in model complexity and are forced to apply approximate inference techniques instead of exact algorithms. Kou et al. [131] and Krishnan et al. [134] have shown that stacked graphical models with exact inference can compete with the accuracy of those complex models. They reduce the computational cost by applying an ensemble for two linear-chain CRFs where they aggregate the output of the first models in order to provide information about related instances or entities to a stacked model.

Although these approaches cannot be applied for exploiting context-specific consistencies, they greatly influenced the development of the machine learning approaches of this work. The approach of Stacked Conditional Random Fields in Section 6.2 was inspired by the stacked graphical learning of Kou et al. [131] and the approaches towards higher order models by the work of Sutton and McCallum [189] about skip-chain Conditional Random Fields.

Chapter 4

UIMA Ruta

UIMA Ruta (Rule-based Text Annotation)[35] is a rule-based tool that focuses especially on the rapid development of information extraction and even general text processing applications. The system consists of a rule language extended with scripting elements and a strong development support. The rule language was designed to provide a compact and comprehensible representation of patterns over annotations without restricting its expressiveness or area of use. It covers almost all features of related rule languages for information extraction while still introducing a few new ones. Although the rule language was optimized for rapid development and not for a fast runtime performance, it yet compares well with similar rule-based systems concerning speed. The UIMA Ruta Workbench provides a full-featured development environment for the UIMA Ruta language, which exceeds the functionality of most related tools[36].

The chapter is structured as follows: Section 4.1 provides an overview of the system and examines its history. The rule-based scripting language is introduced in Section 4.2. The syntax, semantics and the rule matching algorithm are explained in detail and illustrated with examples. Furthermore, a selection of special features and different engineering approaches complete the description of the language. The development environment and available tooling for improving the engineering experience are in focus of Section 4.3. Section 4.4 concludes the chapter with a comparison of UIMA Ruta to related systems.

4.1 Introduction

UIMA Ruta is a system for rule-based information extraction, which is built completely on UIMA. Two major parts can be distinguished: The compact, powerful and extensible rule-based scripting language for arbitrary text processing tasks and the full-featured development environment enriched with various tooling in order to ease and improve the development of rule-based applications.

The UIMA Ruta language is referred to as a rule-based scripting language since it pursues different approaches than common rule-based systems for information extraction and it provides many language elements rather attributed to scripting languages. A rule of the UIMA Ruta language is composed of a sequence of rule elements, which specify a sequential pattern over annotations or text fragments. A rule element essentially consists of a mandatory match reference,

[35] An early version was published under the name TextMarker [122]

[36] The content of this chapter is taken from Peter Kluegl, Martin Toepfer, Philip-Daniel Beck, Georg Fette, Frank Puppe. UIMA Ruta: Rapid Development of Rule-based Information Extraction Applications. Natural Language Engineering, 2014 [125]. Some examples are taken from the documentation of UIMA Ruta written by the same author.

an optional quantifier, an optional list of conditions and an optional list of actions. The match reference creates a connection to the document by matching on the covered text of an annotation of a given type if possible. The quantifier states if the rule element is mandatory and how often the rule element is allowed to match. Conditions provide additional constraints for the text passage matched by the match reference. If all rule elements of the rule and their match references have successfully matched, then the actions of all rule elements are applied creating new annotations or performing different operations. The language was designed to provide a compact and comprehensible representation of patterns over annotations without restricting its expressiveness or area of use. Due to the large amount of different language elements, the language can be utilized to solve various tasks besides its envisaged purpose. These include the specification and execution of pipelines of UIMA components, the definition of UIMA type systems, or other applications like a rule-based document sorter. The UIMA Ruta rule language is described in Section 4.2.

The UIMA Ruta Workbench provides a full-featured development environment for the UIMA Ruta language. It was developed to ease every step in engineering rule-based applications and provides in addition to classical editing support like syntax checking also various different tooling. The system can generate a complete explanation of each step of the rule inference, which enables the rule engineer to adapt rules responsible for unintended behavior. The engineering process is supported by tools for introspection, test-driven development, automatic back-testing, constraint-driven evaluation and automatic rule induction. The interaction of the provided tooling results in a rapid and agile development of well-maintained rule sets. Section 4.3 addresses the features of the UIMA Ruta Workbench.

From the perspective of the UIMA framework, the UIMA Ruta rules are interpreted and executed by a generic analysis engine. UIMA Ruta rules can, therefore, be seamlessly integrated in UIMA applications. The system provides additionally several analysis engines, which can be helpful in various text processing tasks. To these belong, for example, components for annotating HTML documents or for cutting annotated documents while recalculating the offsets of the annotations. The UIMA Ruta Workbench is able to automatically generate descriptor files for rule sets and type system definitions.

4.1.1 History and Current State

The UIMA Ruta language and Workbench was initially developed and published under the name TextMarker. The earliest predecessor that shared some minimal concepts and language keywords has been created by Patrick von Schoen in his diploma thesis [198]. In 2007, a new system has been implemented and greatly extended, whereby it was built on the UIMA framework and provided an Eclipse-based development environment [119]. From 2008, it was hosted at SourceForge until it was contributed to Apache UIMA in 2011. Since then, the rule inference has been reimplemented adding many new language constructs and support for elements of the UIMA framework. The first version of the system was released at Apache UIMA in March 2013. In May 2013, it was renamed to UIMA Ruta (RUle-based Text Annotation).

The system has been actively developed since its contribution to Apache UIMA and introduced a useful tool for rule-based information extraction in the ecosystem of UIMA. The website of

UIMA Ruta[37] provides links to the download section, the documentation and the update site for the UIMA Ruta Workbench.

4.2 The Rule-based Scripting Language

The UIMA Ruta language is primarily a rule-based language for specifying patterns over annotations and additional consequences in the case the pattern matched successfully on a text position. The patterns are applied successively in the order the user has specified them. In order to support more use cases, the language was incrementally extended with elements rather unknown by rule languages, for example, control structures or variables. These two characteristics led to a perspective in which the UIMA Ruta language can be interpreted as a scripting language. This section provides an introduction in the UIMA Ruta rule-based scripting language.

4.2.1 Provided Annotation Types

Before the syntax of the UIMA Ruta language is introduced, this section gives a short overview of UIMA types provided by the system because these types are partially utilized for example in the following sections. For improved readability, only the short names without the namespace of the type are used.

If not configured otherwise, the system adds annotations for different classes of tokens to the document in order to facilitate rapid prototyping. These annotations can be used to define some initial rules, which create for their part new annotations that will be used by other rules. Figure 4.1 contains a summary of these types displayed as a tree structure representing their type inheritance. The leaves of the tree correspond to the types of annotations that will actually be added to the document and are highlighted with a blue background color. Most of the type names are self-explaining: An annotation of the type CW is created for each word starting with an upper-case letter, whereas the type SW is used for lower-cased words. Red and green types serve only for generalization, for example, if the capitalization of a word is irrelevant. Note that also types for whitespaces are provided. The process of adding initial annotations is called seeding and creates a complete partitioning of the document in order to cover each text position with exactly one of those annotation.

Another special type is Document, which can be used synonym to the DocumentAnnotation type of the UIMA framework. It always refers to the complete visible document. The UIMA Ruta language provides language elements for restricting the window. The type Document will in this case refer to the surrounding window instead of the complete document. Other types are either used for the internal rule inference or by the tooling of the UIMA Ruta Workbench. To these belong types that store information about the rule inference for further explanation or profiling.

4.2.2 Syntax and Semantics

The definition of the syntax (grammars) of the rule language is given in Backus-Naur form, which is described by McCracken et al. as following:

[37] `http://uima.apache.org/ruta.html`

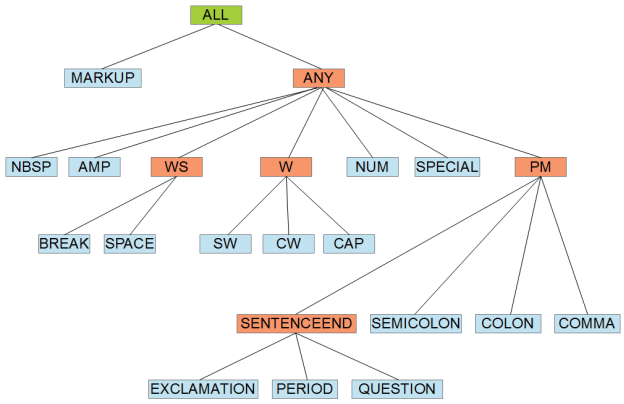


Figure 4.1: Types of initially provided annotations

> "*Backus-Naur Form, named after John W. Backus of the US and Peter Naur of Denmark, and usually written BNF, is the best-known example of a meta-language (q.v.), i.e. one that syntactically describes a programming language. Using BNF it is possible to specify which sequences of symbols constitute a syntactically valid program in a given language.*" [154]

A BNF specification consists of a set of derivation rules which state how elements of the language can be generated. This is done by replacing the nonterminal symbols in angle brackets in the right part by the left part until only terminal symbols remain. In the following grammars of this section, the derivation rules for some nonterminal symbols are neglected in order to improve the readability and to grant a comprehensive overview. The exact syntax for these constructs can be found in the documentation of UIMA Ruta. Additionally, three simplifications of the BNF specification are applied: Optional elements are represented by an appended question mark. Repetitions are indicated by an appended "*" (optional) or "+" (mandatory). Parentheses group alternatives or sequences of elements.

The following sections describe the actual executable resource, the script file, before the syntax of rules in the UIMA Ruta language is defined. Along with the grammars, examples for rules and scripts are given in order to illustrate valid excerpts of the UIMA Ruta language. These listings utilize the same syntax coloring as the editor of the UIMA Ruta Workbench: Keywords for package specification, declarations and imports are depicted with dark red and bold font. Comments are light green and string literals are light blue. Conditions are highlighted with a dark green and bold font, and action with a dark blue and bold font. Given types provided by the UIMA Ruta language for the set of initial annotations are bold with a grey color.

4.2.2.1 Script Definition

The general syntax of a UIMA Ruta script file is given in Grammar 4.1, which states that a script consists of an optional specification of the package followed by an optional list of import declarations and an optional list of statements. An empty document file is, therefore, a valid script file without functionality. The package starts with the keyword "PACKAGE" followed by an identifier that serves, together with the file name of the script, as the namespace of elements defined within the script. The language supports four different kinds of imports: The keyword 'TYPESYSTEM' indicates the import of a UIMA type system descriptor whereby its defined types are made available in the script file. The keyword 'SCRIPT' includes an additional script and its known types for further usage. The remaining two keywords import analysis engines, which can then be executed from within the script file. Following the imports, a list of statements constitutes the major part of the script file and its actual functionality. Three different groups of statements can be distinguished. Declarations define new UIMA types, new variables or external dictionaries. The definition of new types only serves for rapid development in the UIMA Ruta Workbench since it avoids switching to other tools. The other two declarations, however, define new elements of the UIMA Ruta language itself. The block statement is a script-like control structure that provides special functionality for the user such as procedures, definition by cases or restriction of the window the rules are applied in. This construct and similar language elements are described in Section 4.2.5. The last kind of statement, the rules, provides the actual functionality of the script file and is described in the next section after an example of a common composition of a script file.

⟨*script*⟩ ::= ⟨*package*⟩? ⟨*import*⟩* ⟨*statement*⟩*

⟨*import*⟩ ::= ('TYPESYSTEM' | 'SCRIPT' | 'ENGINE' | 'UIMAFIT')
⟨*identifier*⟩ ';'

⟨*statement*⟩ ::= ⟨*declaration*⟩ | ⟨*rule*⟩ | ⟨*block*⟩

⟨*declaration*⟩ ::= ⟨*TypeDeclaration*⟩
| ⟨*VariableDeclaration*⟩
| ⟨*DictionaryDeclaration*⟩

⟨*block*⟩ ::= 'BLOCK' '(' ⟨*identifier*⟩ ')' ⟨*ruleElement*⟩ '{' ⟨*statement*⟩+ '}'

Grammar 4.1: Simplified grammar of the script syntax.

Listing 4.1 contains an example with diverse language elements. The example starts with a specification of the package of the script file, which is applied for defining the namespace of newly defined types and blocks. In the lines 3 and 17, two comments provide an explanation of the corresponding environment. From line 4 to line 8, different global imports are added starting by importing a type system descriptor and its type definitions. Then, three additional scripts are included and finally an external analysis engine is imported in line 8. Line 10 contains a type declaration, whereas only the short name of the type is given. The following lines provide simple

rules, which execute the imported analysis engine and script files on the current document. From line 18 finally, a block declaration applies a rule for each annotation of the type Reference. This example illustrates just one script-like feature of the UIMA Ruta rule language since it actually defines a pipeline of different components with some additional post-processing. Solving different engineering tasks like the combination of components or the declaration of new types is an important aspect for the rapid development and avoids switching to other tools.

```
1  PACKAGE uima.ruta.example;
2
3  // import the types of this type system
4  TYPESYSTEM types.BibtexTypeSystem;
5  SCRIPT uima.ruta.example.Author;
6  SCRIPT uima.ruta.example.Title;
7  SCRIPT uima.ruta.example.Year;
8  ENGINE uima.ruta.example.SegmentationEngine;
9
10 DECLARE Reference;
11 Document{-> CALL(SegmentationEngine)};
12
13 Document{-> CALL(Year)};
14 Document{-> CALL(Author)};
15 Document{-> CALL(Title)};
16
17 // create bibtex annotation
18 BLOCK(forEach) Reference{} {
19         Document{-> CREATE(Bibtex, "author" = Author,
20                 "title" = Title, "year" = Year)};
21 }
```

UIMA Ruta Listing 4.1: UIMA Ruta script pipeline for parsing bibliographic references.

4.2.2.2 Rule Definition

The syntax of rules in the UIMA Ruta language is specified in Grammar 4.2. A rule commonly consists of a sequence of rule elements followed by a semicolon indicating the end of the rule. Simple regular expression rules serve to create additional annotations and assign feature values based on the matching groups of a regular expression. They have been added to the UIMA Ruta language additionally to the normal rules in order to support rapid prototyping. The second kind of special rules, conjunctions of rules, will be discussed in Section 4.2.3.3. A rule element itself consists at least of the mandatory match reference, which creates a connection to the document by matching on a text fragment. This connection can be specified by five different kinds of references. A type expression represents a UIMA type and is the most common kind of match reference. The rule element matches on the annotation of the given type and, therefore, on the position covered by the annotation. The feature expression is an extension of the type expression by adding additional constraints on the feature values of the matched annotation or by referring to the annotations stored as feature values. A string expression represents a

⟨*rule*⟩ ::= (⟨*ruleElement*⟩+ | ⟨*regExpRule*⟩ | ⟨*conjunctRules*⟩) ';'

⟨*ruleElement*⟩ ::= '@'? ⟨*matchReference*⟩ ⟨*quantifier*⟩? ('{' ⟨*conditions*⟩?
'->' ⟨*actions*⟩? '}')? ⟨*inlinedRules*⟩?

⟨*matchReference*⟩ ::= ⟨*typeExpression*⟩ | ⟨*stringExpression*⟩
| ⟨*featureExpression*⟩ | ⟨*composedRE*⟩ | ⟨*wildCard*⟩

⟨*composedRE*⟩ ::= '(' ⟨*ruleElement*⟩+ ')'
| '(' ⟨*ruleElement*⟩ ('&' ⟨*ruleElement*⟩)+ ')'
| '(' ⟨*ruleElement*⟩ ('|' ⟨*ruleElement*⟩)+ ')'

⟨*quantifier*⟩ ::= '?' | '??' | '*' | '*?' | '+' | '+?'
| '[' ⟨*numberExpression*⟩ ',' ⟨*numberExpression*⟩ ']'
| '[' ⟨*numberExpression*⟩ ',' ⟨*numberExpression*⟩ ']' '?'

⟨*conditions*⟩ ::= ⟨*condition*⟩ (',' ⟨*condition*⟩)*

⟨*actions*⟩ ::= ⟨*action*⟩ (',' ⟨*action*⟩)*

⟨*condition*⟩ ::= `ConditionKeyword` '(' ⟨*expression*⟩ (',' ⟨*expression*⟩)* ')'

⟨*action*⟩ ::= `ActionKeyword` '(' ⟨*expression*⟩ (',' ⟨*expression*⟩)* ')'

⟨*inlinedRules*⟩ ::= ('->' | '<-') '{' ⟨*rule*⟩ + '}'

Grammar 4.2: Simplified grammar of the rule syntax.

character sequence and enables the rule element to match directly on the text passage of the document with the identical character sequence. The match reference of a rule element can also consist of rule elements creating a sequential grouping, and conjunctive or disjunctive constraints. The last kind of match reference is the wildcard, which provides a placeholder for any kind of text passage. The wildcard is described in Section 4.2.3.

A rule element can be extended with several optional language elements. The anchor marker "@" in front of a rule element is ignored for now and is discussed in Section 4.2.3. The optional quantifier part (Kleene operator) specifies how often the rule element may or has to match for a successful rule match. The UIMA Ruta language supports the four most common quantifiers known by regular expressions, each in a greedy and reluctant form. Greedy quantifiers terminate only if the rule element failed to match and reluctant quantifiers complete their repetition already if the next rule element is able to match. Quantifiers that restrict the minimal and maximal amount of repetitions enclose these values in square brackets. A rule element without a quantifier has to match exactly once. A rule with several rule elements and optional quantifiers, but without any other elements specifies a sequential pattern over annotations or character sequences.

Before the remaining elements of a rule are described, a few examples of patterns specified with UIMA Ruta are given. The example in Listing 4.2 contains a rule with two rule elements, each consisting only of a match reference. The first rule element matches on the token "Room" and the second one on an annotation of the type "NUM" that is a sequence of digits. The rule matches successfully on all positions where the token "Room" is followed by a number.

```
1 //... matches on "Room 123"
2 "Room" NUM;
```

UIMA Ruta Listing 4.2: Simple UIMA Ruta rule with two rule elements.

Listing 4.3 provides a rule with three rule elements whereas the second rule element is optional and may match repeatedly. This rule matches on all positions that are annotated with an article followed by a noun, and an arbitrary amount of adjectives may by present between both annotations.

```
1 //... matches on "the blue bird"
2 ART ADJ* NN;
```

UIMA Ruta Listing 4.3: Simple UIMA Ruta rule with three rule elements.

The rule in Listing 4.4 contains two rule elements, one with a composed match reference and one with a simple type expression. The first rule element may match repeatedly due to its quantifier. Hence, the rule matches on a list of annotations of the type animal separated by commas.

```
1 //... matches on "Dog, Cat, Bird"
2 (Animal COMMA)+ Animal;
```

UIMA Ruta Listing 4.4: Simple UIMA Ruta rule with a composed rule element.

After the optional quantifier, curly brackets contain lists of optional conditions and actions, which are separated by a rightwards arrow "->". Conditions are binary predicates and specify additional constraints on the matched position that need to be fulfilled for a successful match of the rule element. Actions represent the consequence of the rule and are only applied if the complete rule was able to match successfully. Conditions and actions start both with a keyword indicating its type followed by a list of expressions in parentheses determining possible arguments and configurations. The last optional part of a rule element, the inlined rules, is discussed in Section 4.2.5. Examples of rules extended with additional conditions and actions are given in the following. In contrast to the previous examples, these rules have actual consequences when they match successfully.

The rule in Listing 4.5 consists of three rule elements. The first one (ANY...) matches on every token, which has a covered text that occurs in a word list `MonthsList`. The second rule element (PERIOD?) is optional and does not need to be fulfilled, which is indicated by

```
1 //... matches on "Dec. 2004", "July 85" or "11.2008"
2 ANY{INLIST(MonthsList) -> MARK(Month), MARK(Date,1,3)}
3     PERIOD? NUM{REGEXP(".{2,4}") -> MARK(Year)};
```

UIMA Ruta Listing 4.5: A rule with additional actions and conditions.

its quantifier "?". The last rule element (NUM...) matches on numbers that fulfill the regular expression REGEXP(".{2,4}") and are therefore at least two characters to a maximum of four characters long. If this rule successfully matches on a text passage, then its three actions are executed: An annotation of the type Month is created for the first rule element, an annotation of the type Year is created for the last rule element, and an annotation of the type Date is created for the span of all three rule elements. If the word list contains the correct entries, then this rule matches on strings like "Dec. 2004", "July 85" or "11.2008" and creates the corresponding annotations.

```
1 Paragraph{CONTAINS(Bold, 90, 100, true),
2     CONTAINS(Underlined, 90, 100, true), ENDSWITH(COLON)
3     -> MARK(Headline)};
```

UIMA Ruta Listing 4.6: A rule with additional actions and conditions.

The match reference of the rule element in Listing 4.6 is given with the type "Paragraph", thus the rule investigates all Paragraph annotations. The rule matches only if the three conditions are fulfilled. The first condition CONTAINS(Bold, 90, 100, true) states that 90%-100% of the matched annotation should also be annotated with annotations of the type Bold. The boolean parameter "true" indicates that amount of Bold annotations should be calculated relatively to the matched annotation. The two numbers "90,100" are, therefore, interpreted as percent amounts. The second condition CONTAINS(Underlined, 90, 100, true) consequently states that the paragraph should also contain at least 90% of annotations of the type "underlined". The third condition ENDSWITH(COLON) finally forces the Paragraph annotation to end with a colon. It is only fulfilled, if there is an annotation of the type COLON, which has an end offset equal to the end offset of the matched Paragraph annotation.

The conditions and actions in the previous examples reflect only a small sample of the available constructs. The language provides more than 25 different conditions and more than 40 different actions, which enables the knowledge engineer to tackle different annotation tasks efficiently. An overview of the available actions and conditions is given in Figure 4.2. Actions, for example, can not only add new annotations, but are also able to remove (UNMARK) or modify existing ones (SHIFT), or execute other components (EXEC). Special actions are also applied for creating relations between entities or simply copying feature values as shown in Example 4.7.

The rule creates a new annotation of the type Container for each Token annotation. Furthermore, three features are automatically filled with values: The value of the feature "posTag" of the Token annotation is assigned to the feature "pos" of the Container annotation. The value of the feature "value" of a Lemma annotation with the same offsets as the matched Token annotation

is assigned to the feature "lemma" of the Container annotation. Finally, the complete Token annotation is stored in the feature "token".

The language also supports syntactic sugar that allows one to specify conditions and action using expressions without keywords. The user can use boolean expressions, such as boolean variables, or feature-match expressions in order to formulate compact conditions. As for actions, a type expression is able to replace a MARK action and feature-assignment expressions are able to modify the values of matched feature structures. Listing 4.8 provides an example how the rule of Listing 4.5 can be rewritten without MARK actions.

The conditions and actions are normally clearly separated in well-known rule languages. While the conditions build the left-hand part, the actions follow in a right hand part after a distinctive separator. In the UIMA Ruta language, actions can occur at each rule element after the list of conditions and are not located solely at the end of the rule as it would be expected. Allowing action to be attached to the rule element that provides the context for its consequences provides various advantages and results in a more compact rule representation. However, the actions can be detached and listed at the last rule element in most situations, simulating the traditional composition. Listing 4.9 provides an example how the rule of Listing 4.5 can be rewritten with the actions located at the end of the rules.

4.2.2.3 Extensible Language Definition

The UIMA Ruta language provides a variety of actions and conditions, which can be applied in order to effectively address various tasks. Different users have, however, different use cases and require specialized language constructs for efficiently implementing their rule-based application. UIMA Ruta provides, therefore, a concept for extending the language by additional actions, conditions, functions and even blocks that change the execution of rules. The user is not restricted to the available list of language elements, but can adapt and optimize the language from the perspective of her use cases.

4.2.3 Inference

The description of the syntax and semantics in the last section provided an overview of the specification of valid rules and their meaning. This section now considers how the rules are applied. Before the matching algorithm is introduced, the order of rule application is described.

4.2.3.1 Rule Execution

UIMA Ruta rules are applied in an imperative manner, one rule after each other, in the order they occur in the script file. This leads to an interpretation of the language as cascaded finite-state transducers, whereas each transducer corresponds to one rule. Other rule-based languages for information extraction compile the rules of one phase into one finite-state transducer in order to avoid unnecessary and duplicate inference steps. The execution order of UIMA Ruta is rather known by programming or scripting languages and is one reason why the UIMA Ruta language is referred to as a rule-based scripting language. Although this kind of inference provides some disadvantages, the advantages prevail in the focus of the system, which is rapid development of rule-based information extraction applications.

Conditions	AFTER, AND, BEFORE, CONTAINS, CONTEXTCOUNT, COUNT, CURRENTCOUNT, ENDSWITH, FEATURE, IF, INLIST, IS, LAST, MOFN, NEAR, NOT, OR, PARSE, PARTOF, PARTOFNEQ, POSITION, REGEXP, SCORE, SIZE, STARTSWITH, TOTALCOUNT, VOTE
Actions	ADD, ADDFILTERTYPE, ADDRETAINTYPE, ASSIGN, CALL, CLEAR, COLOR, CONFIGURE, CREATE, DEL, DYNAMICANCHORING, EXEC, FILL, FILTERTYPE, GATHER, GET, GETFEATURE, GETLIST, GREEDYANCHORING, LOG, MARK, MARKFAST, MARKFIRST, MARKLAST, MARKONCE, MARKSCORE, MARKSCORE, MARKTABLE, MATCHEDTEXT, MERGE, REMOVE, REMOVEDUPLICATE, REMOVEFILTERTYPE, REMOVERETAINTYPE, REPLACE, RETAINTYPE, SETFEATURE, SHIFT, TRANSFER, TRIE, TRIM, UNMARK, UNMARKALL

Figure 4.2: List of conditions and actions currently available in UIMA Ruta.

```
Token{-> CREATE(Container, "pos" = Token.posTag,
  "lemma" = Lemma.value, "token" = Token)};
```

UIMA Ruta Listing 4.7: A simple rule for copying feature values and assigning annotations to features. A new annotation of the type Container is created, which stores different information of the underlying annotations as feature values.

```
(ANY{INLIST(MonthsList) -> Month} PERIOD?
        NUM{REGEXP(".{2,4}") -> Year}){-> Date};
```

UIMA Ruta Listing 4.8: The rule of Listing 4.5 without MARK actions.

```
ANY{INLIST(MonthsList)} PERIOD? NUM{REGEXP(".{2,4}")
        -> MARK(Month,1), MARK(Year,3), MARK(Date,1,3)};
```

UIMA Ruta Listing 4.9: The rule of Listing 4.5 with a clear separation of conditions and actions.

The disadvantages consist mainly in the possibility of a decreased performance for large rule sets and in the missing of truth maintenance between rules. However, truth maintenance is hardly supported by automata-based languages in general. The performance issue compared to rules compiled in one single finite-state transducer occurs especially if many rules start with the same match reference resulting in a variety of identical and redundant operations. If, however, the rules have no joint match references, the performance differs only marginally since the rules share no states in the automata. The absence of truth maintenance leads potentially to an increased amount of rules. If a rule activates or negates the preconditions of previous rules, then their postconditions are not automatically executed or revoked. The user has to add additional rules for propagating the desired effect like in other languages.

The advantages of this kind of imperative rule execution lie first of all in its simplicity, which is important for rapid development of rule sets and can be essential for inexperienced users or users not familiar with rule-based systems. The user does not have to consider side effects to previous rules and snares of dependent rules. Due to the missing truth maintenance, the rules of a script have to be ordered corresponding to their dependencies, which results in clear and comprehensive rule sets. The absence of a truth maintenance implies also in an improved performance, because the match references and conditions of other rules do not need to be reevaluated after a rule added or removed annotations. Furthermore, the ability to revoke the postconditions of rules either confines the rule language or increases its complexity. The usage of rule scripts with truth maintenance as a prototyping language for the definition of pipelines, for example, is only possible if the operations of arbitrary components can be taken back. The linear execution of rules even makes the definition of pipelines possible in the first place. Another advantage consists in an easier explanation of the rule inference, which enables the user to quickly identify the causes of undesired rule behavior.

The use of the UIMA Ruta language in practical and real-world scenarios has shown that the advantages outweigh the disadvantages, especially if the focus lies on the rapid and efficient development of rule-based information extraction applications. There have been plans to introduce a language construct similar to the BLOCK environment, which compiles the contained rules into an automata, but this extension of the language and inference remains for future work.

4.2.3.2 Rule Matching

Rules in UIMA Ruta are atomic statements concerning the inference, as pointed out in the last section. A rule itself can be interpreted as an automaton with states for the match references and conditions, and with state transitions between the different rule elements. For the description of the rule matching in UIMA Ruta, a pseudo code algorithm is employed.

The rule matching of UIMA Ruta is specified in Algorithm 2. The rule starts to match by calling the procedure startMatch. First, the rule element is determined, which starts the rule matching process. This is normally the first rule element of the rule resulting in a left-to-right match. If the match reference of the selected rule element consists of a composed rule element, then the starting rule element is determined in the list of contained elements. The starting rule element is then requested to continue the matching process with the procedure continueMatch.

The procedure continueMatch provides one to three arguments: the current rule element, an optional position and an optional rule match. If this method is called by the procedure

Algorithm 2 Simplified pseudo-code of the rule matching algorithm in UIMA Ruta.

```
procedure startMatch
    rule element e ← identify starting rule element
    continueMatch(e)
end procedure

procedure continueMatch(rule element e, optional position p, optional rule match o)
    if position p is given then
        anchors a ← all valid positions next to position p
    else
        anchors a ← all valid positions for the rule element e
    end if
    for all positions i of anchors a do
        rule match m ← new alternative of rule match o or new rule match
        validate match reference and conditions of e on position i for rule match m
        rule element n ← nextElement(rule element e, rule match m)
        continueMatch(rule element n, position i, rule match m)
    end for
end procedure

function nextElement(rule element e, rule match m)
    if quantifier of e indicates further repetition then
        return rule element e
    else
        rule element n ← identify rule element next to e
        if rule element n exists and rule match m is valid then
            return rule element n
        else
            doneMatching(rule match m)
        end if
    end if
end function

procedure doneMatching(rule match m)
    if rule match m is valid then
        apply all actions of rule
    end if
end procedure
```

startMatch, then only one argument is given. First of all, valid positions for the rule element are determined. Either all valid positions, e.g., annotations of the type of the match reference, are listed, or only valid positions next to the provided position in the arguments. The strategy for selecting the next position can be configured. For each possible position of this list, several operations are performed. First, a new rule match is created, which stores the current state of the matching process, e.g., already evaluated positions. If a rule match was provided by the arguments of the procedure, then a copy is created, representing a new alternative rule match. The position is validated concerning the match reference, conditions and inlined rules of the rule element. UIMA Ruta is, therefore, able to handle positions where multiple annotations begin. If the rule element has successfully matched, then the next rule element is identified using the function nextElement. The next rule element is then requested to continue the matching process on the new position and rule match.

The function nextElement does not only provide the next rule element, but also terminates the matching process, if no remaining rule elements can be found. The function initially checks whether the quantifier of the rule element allows repetitions, and returns the current one as appropriate. Reluctant and optional quantifiers are neglected in the pseudo code for simplicity. If the rule element has already matched often enough, then the next rule element is determined. This is normally the rule element following the current one. If the rule element is part of a composed rule element, then next rule element can also be the composed one. The last procedure doneMatching simply validates if the current rule match was successful and then applies all actions and inlined rules of the rule on the positions stored in the rule match. The matching algorithm tracks a rule match until it terminates and possibly applies the actions before an alternative match is considered. This facilitates the specification of useful rules. Two examples illustrate the consequences of this feature.

```
1 ANY+{-> Text1};
2 ANY+{-PARTOF(Text2) -> Text2};
```

UIMA Ruta Listing 4.10: Example for dependencies between rule matches.

Listing 4.10 contains two rules. The first rule matches on each token of the document and creates an annotation of the type Text1 for the covered positions. However, this is caused only by the first iteration in the procedure continueMatch. The next iteration considers the second token of the document and continues the matching process and so on. This results in annotations starting at each token and ending at the end of the document. Although this behavior seems to be counterintuitive, there are situations where this kind of matching is desired. The second rule provides an additional condition, which states that the matched position is not allowed to be part of an annotation of the type Text2. This rule, therefore, creates only one annotation covering the complete document. The user is also able to influence the matching process by configuring that matched positions should not be considered by the same rule.

```
1 PERIOD Annotation{-> BetweenPeriods} PERIOD;
```

UIMA Ruta Listing 4.11: Example for potential alternatives at one position.

The rule in Listing 4.11 creates an annotation of the type BetweenPeriods for all annotations that are surrounded by periods. If several annotations start after a period, e.g., a token, a sentence and a paragraph, then the largest annotation is matched first and the matching process continues with the last rule element. The second largest annotation is considered only then, after the rule already potentially created an annotation.

The description of the matching algorithm mentioned that the starting rule element is normally the first rule element of the rule, which can decrease the performance of the matching process for certain rules to some extent. This problem is illustrated with two rules in Example 4.12 that match on the second last token.

```
1 ANY LastToken;
2 ANY @LastToken;
```

UIMA Ruta Listing 4.12: Two simple rules that match on a token followed by a LastToken annotation. While the first rule has to investigate every token, the second rule starts to match with the second rule element and requires less index operations.

The first rule investigates each token of the document starting with the first one, which results in many unnecessary rule matches. The second rule provides a start anchor, indicated by the symbol "@". This optional symbol enables the user to manually specify the starting rule element. The rule begins with the annotation of the type LastToken and continues the match with the previous rule element, which results in a right-to-left matching and in only one rule match. The UIMA Ruta language also provides an option called "dynamic anchoring" that automatically determines the starting element using a heuristic applied on the amount of involved annotations. The option can be activated in the configuration parameters or by other rules in the script. The possible occurrences of matching references of the rule elements are compared and the rule element with the least amount of initial matches is selected. The heuristic also includes the quantifier and composed rule elements. Furthermore, a penalty for the reverse matching direction can be specified. Dynamic anchoring is a newer feature and thus not yet activated by default.

The special kind of match reference, the wildcard "#", can be used to further optimize the rule matching performance. The two rules in Listing 4.13 create an annotation of the type Sentence for all text passages that are surrounded by periods. While the first rule matches on one token after each other until the next occurrence of a period, the second rule directly matches on the next period. This reduces the amount of considered positions and also provides a compact representation.

```
1 PERIOD ANY+?{-> Sentence} PERIOD;
2 PERIOD #{-> Sentence} PERIOD;
```

UIMA Ruta Listing 4.13: Two equivalent rules for annotating text between two periods. While the first rule needs to match on each token (ANY), the second rule just searches for the next period resulting in less UIMA index operations.

The general performance concerning execution time of the UIMA Ruta rule-based script mainly depends on the amount of index operations in the UIMA framework, and related to this, on the amount of rule matches and the involved conditions. While those operations of the rule matching algorithm are easily estimated for a given set of rules and the documents they are applied on, additional index operations can be performed by the conditions and actions. One example is the condition NEAR, which evaluates the nearby presence of certain types of annotations independently of the match references of adjacent rule elements. In order to achieve this, the condition has to investigate the annotation index in relation to the currently matched position.

The language supports many different ways to specify an annotation problem. Similar to programming languages, it is possible in UIMA Ruta to implement slow and fast rule sets for solving the same problem. Since the language was especially designed to support rapid development, the user normally does not stress performance issues in the first place. If, however, the execution time of a rule script needs to be improved, many different ways exist. This section introduced a few options to reduce the amount of rule matches or index operations. The user can also make use of efficient dictionaries or can profile the rule execution in order to identify bottlenecks (cf. 4.3.2).

4.2.3.3 Beyond Sequential Matching

The matching process described until now considers only rule elements as sequential patterns. The language also supports rule elements that need to match on the same position. As introduced in Section 4.2.2.2, composed rule elements are not only able to specify sequential patterns, but also disjunctive and conjunctive rule elements. The list of disjunctive rule elements separated by the symbol "|" specify that at least one of the rule elements needs to match for a successful match of the composed rule element. Analogously, all of the conjunctive rule elements separated by the symbol "&" need to match in order to continue the matching process. These language elements can be applied for verifying different aspects on the same position that cannot be represented by a combination of conditions.

Another language construct that does not represent a strict sequential pattern is given by the symbol "%", which is applied to connect two rules. The resulting rule builds a conjunction of both rules and its actions are only applied if both rules successfully matched. The connected rules themselves may match independently of each other on positions in the current window or document.

```
1 // matches on ''Room 1 ... ...  call 911'';
2 CW NUM % SW NUM;
```

UIMA Ruta Listing 4.14: A conjunction of two simple rules. The complete rule matches only if both rules are able to match independently of each other.

The rule in Listing 4.14 consists of two connected rules and matches only successfully if there is an arbitrary capitalized word that is followed by a number in the document and if there is also some arbitrary lower-cased word that is followed by a number.

4.2.4 Visibility and Filtering

The UIMA Ruta language and its inference are designed to provide an instrument for solving different text processing and information extraction tasks. One step in this direction is the possibility of defining patterns not only over token but over arbitrary types of annotations. Rules can, therefore, be applied for matching sequences of tokens, but also sequences of paragraphs. Another feature provided by UIMA Ruta is the specification of the visibility that determines which kinds of text passages represented by annotations are accessible by the rules. While one type of text is important in one use case, it should be ignored in other applications. If the rules are applied to, for example, label sequences of tokens, then the whitespaces between tokens are of minor interest and the user should be able to ignore them in the definition of the rule set. However, if the rules are built in order to parse identifiers or indentation of tables, whitespaces are essential and need to be included in the pattern. Another example is the processing of headlines in a document. In a sequential pattern over headline annotations, the paragraphs do not need to be considered in the specifications of rules.

Rule-based languages often specify the types available in a phase whereas annotations of other types are automatically skipped. The UIMA Ruta language provides a more complex and dynamic concept of visibility. Here, text positions covered by annotations of specific types are invisible. This leads to a coverage-based visibility concept instead of a type-based one. All annotations and their covered text passages that start or end with a type specified in the set of filtered types are not accessible by the rules. This means that these invisible positions will be skipped when the next position for the rule match is determined. The set is calculated using three lists. The *default* list is specified in the configuration parameters and generally contains types for whitespace and markup. The elements of the *filtered* list are added to this list. Afterwards, the elements of the *retained* list are removed. The *filtered* and *retained* lists can be modified by actions so that the knowledge engineer is able to adapt the rule inference to her current requirements directly in a UIMA Ruta script. The exact behavior of rules facing invisible annotations is explained with Example 4.15 and Example 4.16.

```
1 // matches on ``Dec<br>2004'', ``May1999'' or ``July 85''
2 W NUM;
3 Document{-> RETAINTYPE(SPACE,MARKUP)};
4 // matches only on ``May1999''
5 W NUM;
```

UIMA Ruta Listing 4.15: Two identical rules that match on different text positions due to the changed filtering settings in the second rule. The second rule is sensible to markup and whitespaces in its sequential constraint.

The first rule in Listing 4.15 matches on text fragments like "Dec
2004", "May1999" or "July 85" since whitespaces and markup are filtered by default. The second rule changes the filtering settings by adding the types for space and markup annotations to the *retained* list, which makes them visible again. The last rule is identical to the first one, but matches now only on the text fragment 'May1999" since the matching process was unable to find a valid subsequent position.

```
1 Sentence;
2 Document{-> RETAINTYPE(MARKUP)};
3 Sentence;
4 Document{-> FILTERTYPE(Headline)};
5 Sentence;
6 Document{-> RETAINTYPE, FILTERTYPE};
```

UIMA Ruta Listing 4.16: Three rules for matching on sentences. The other rules change the filtering setting resulting in different matches on sentences.

The Example 4.16 contains three identical rules that match on sentences and three rules that change the filtering settings. The first rule matches on sentences that do not start with a whitespace or markup annotation due to the default filtering settings. The second rule enables matching on markups. The third rule (line 3) matches also on sentences that start with a markup element. The fourth rule hides headlines and, thus, the fifth rule is not able to match on sentences that start with or are part of a headline. The last rule resets the filtering settings to its default values.

4.2.5 Blocks and Inlined Rules

The implementation of annotation tasks using only a plain rule language often leads to rule sets that are hard to interpret by a human. This is caused by the lack of control structures, which are compensated by additional conditions or activation rules. Control structures can be very useful in rule-based information extraction system. Examples for those elements are the restriction of the rule match to a certain window, further modularization, conditioned execution of rule sets or application of rules for each occurrence of an annotation.

The UIMA Ruta language provides the BLOCK construct for these use cases. The syntax of blocks was already specified in Section 4.2.2.1. A block construct starts with the keyword "BLOCK" followed by an identifier that is utilized in case the block is invoked by another rule. The main part, the head, is a rule element, which specifies the functionality of the block. The body of the construct finally contains a list of statements, e.g., rules. The rules within the block are only applied in the context of the matches of the rule element in the head. If the rule element did not match, then the contained rules are not applied at all, which corresponds to a conditioned statement. If the rule element matches on several annotations, then the contained rules are applied once for each matched annotation and only within this annotation, which corresponds to an iteration over the annotations and a restriction of the context. These use cases are illustrated with examples.

```
1 BLOCK(German) Document{FEATURE("language", "de")} {
2     // rules for german documents
3 }
```

UIMA Ruta Listing 4.17: A conditioned statement using the block construct. The contained rules are only applied if the language of the document is set to "de".

Listing 4.17 provides a block construct that applies the contained rules only if the language of the document was set to "de".

```
1 BLOCK(ForEach) Sentence{} {
2     // ... do something
3 }
```

UIMA Ruta Listing 4.18: Iteration over annotations of the type Sentence. The contained rules are applied for each sentence and only in the window of the current sentence.

The block in Listing 4.18 applies the contained rules on each sentence. Rules that try to match over the boundaries of a sentence will automatically fail. Within the body of the block, the type Document refers to the current sentence rather than to the whole document.

Another language element that provides similar functionality are an extension of a rule element, the so called inlined rules (cf. Section 4.2.2.2). It occurs in two manifestations, either interpreted as consequences indicated by the symbol "->" or as preconditions ("<-"). The former kind provides the similar functionality as the block element, but can directly be utilized in more complex rules. If the rule matched successfully, then the inlined rules are applied in the context of the match of the rule element. The latter provides functionality for expressing more complex, nested conditions. Here, the rule itself matches successfully in the first place, if one of the contained rules was able to match. Both extensions enable the user to specify complex patterns in a compact representation. Their syntax and semantics are illustrated with two examples.

```
1 Prefix Sentence->{
2     Document{-STARTSWITH(NP) -> SentNoLeadingNP};
3 };
```

UIMA Ruta Listing 4.19: An example of an inlined rule interpreted as a postcondition. An annotation is created for each sentence if additional requirements are fulfilled.

The rule in Listing 4.19 matches on an annotation of the type Prefix followed by a sentence annotation. If this match was successful, then the rule in line 2 is applied in the context of the matched sentence annotation. It creates an annotation of the type SentNoLeadingNP with the offsets of the matched sentence, if the sentence does not start with an annotation of the type NP.

```
1 Sentence{-> SentenceWithNPNP}<-{
2     NP NP;
3 };
```

UIMA Ruta Listing 4.20: An example of a rule element with an inlined rule interpreted as a precondition. An annotation is created only if the sentence contained two subsequent noun phrases.

Listing 4.20 contains a rule element extended with an inlined rule interpreted as a precondition. The rule tries to match on each annotation of the type Sentence, but only succeeds if this sentence contains an annotation of the type NP followed by another NP.

4.2.6 Engineering Approaches

The support of different engineering approaches for solving an annotation task is an important characteristic of a generic rule-based information extraction system. The formalisation of rules based on only one engineering perspective may lead to inconvenient representations. The UIMA Ruta language was in particular designed to provide a generic pattern formalism that enables the user to solve annotation tasks with different approaches. Among other things, this is achieved with special conditions and, in particular, with actions that encapsulate the necessary functionality. In the following, a selection of approaches are discussed and illustrated with examples. These approaches do not have to be applied separately, but can also be mixed at each stage. The most important approach in the context of this work, engineering for context-specific consistencies, is covered in the next Chapter 5 with some case studies.

4.2.6.1 Classical Approaches

Many classical approaches for rule-based information extraction can be identified. The simplest one is called *candidate classification*, which generates possible candidate annotations and then assigns a specific type, if the candidate holds certain properties. The rule in Listing 4.21, for example, considers each annotation of the type paragraph and labels it as a headline, if it is mostly bold, underlined and ends with a colon.

```
Paragraph{CONTAINS(Bold, 90, 100, true),
    CONTAINS(Underlined, 90, 100, true), ENDSWITH(COLON)
    -> MARK(Headline)};
```

UIMA Ruta Listing 4.21: Candidate classification with UIMA Ruta rules. The rule classifies a paragraph as a headline if it is ninety to hundred percent covered by Bold and Underlined annotations, and ends with a colon.

Interesting annotations can be approached in a *bottom-up* or a *top-down* manner. The former approach starts by identifying small parts of the targeted annotations and successively composes them until the desired annotation can be specified. The latter approach starts with more general annotations and refines or reduces the considered positions until interesting annotations are found. Listing 4.22 provides an example for labeling the author section of a scientific reference using a bottom-up approach. The first rule creates annotations for name initials, which consist only of one upper-case letter followed by a period. The second rule specifies names as a capitalized word followed by a comma and a list of name initials. The last rule finally combines a listing of names to the annotations of the type Author.

```
(CW{REGEXP(".")} PERIOD){-> Initial}; // P.
(CW COMMA Initial+){-> Name}; // Kluegl, P.
(Name (COMMA Name)*){-> Author}; // Kluegl, P., Toepfer, M.
```

UIMA Ruta Listing 4.22: Bottom-up approach for labeling author sections. the first rule detects initials, the second rule identifies names, and the third rule combines names to authors.

The specification of rules that consider the content of an annotation are sometimes hard to define. However, the boundaries of the targeted annotation are possibly easier to identify without facing the variety of patterns occurring within that annotation. An example of this *boundary matching* approach is given in Listing 4.23. The first rule identifies the start of the author part of a reference and the second rule detects possible ends of the author section. The third rule creates an annotation for spans that starts with an annotation of the type AuthorStart and ends with an annotation of the type AuthorEnd.

4.2.6.2 Transformation-based Rules

Transformation-based rules are applied on already existing annotations and try to correct specific errors or defects. The usage of transformations can greatly ease the definition of patterns and accelerate the development of rule sets. The inclusion of all possible exceptions or negative preconditions in the rule that creates the interesting annotations leads to confusing rule constructs. These exceptions can be neglected initially if later transformation-based rules remove annotations that should have been excluded. The transformation-based approach can also be beneficial in many other scenarios. One example is the usage of transformations for domain adaptation or improving the accuracy of arbitrary models.

The UIMA Ruta language provides several actions for modifying existing annotations. The action UNMARK removes annotations of the given type, the action SHIFT changes the offsets of an annotation dependent on the other rule elements and the action TRIM reduces the span of an annotation. Listing 4.24 provides a simple rule that removes annotations of the type Headline if they contain no words at all. A previous rule identified headlines using the layout of the document, but did not include a condition validating their contents. If only one rule was responsible for annotating headlines, the additional precondition is easily added and does not decrease the readability of the rule. If the annotations of the type Headline are, however, created by twenty different rules, then the additional condition has to be added twenty times, which actually decreases the readability and aggravates possible refactoring of the script.

The rule in Listing 4.25 expands the span of an annotation of the type Person, if it is preceded by the word “Mr” and an optional period.

4.2.6.3 Scoring Rules

It is sometimes not possible to specify a combination of properties in one rule. In some situations the knowledge engineer wants to weight different aspects for dealing with uncertainty. The UIMA Ruta language provides a special action and condition for such use cases. The action MARKSCORE adds a heuristic score for a certain kind of annotation and the condition SCORE is able to evaluate this score for further processing. While scoring rules can help to solve problematic tasks in a compact manner, larger sets of scoring rules get increasingly hard to maintain. The author of this work was involved in the development of different techniques for leveraging this problem [12, 8], but these have not been integrated in UIMA Ruta. Example 4.26 contains a small example for the identification of headlines.

The rules from line 2 to line 7 weight different aspects and assign each a score to the annotation HeadlineInd. The first two rules, for example, specify that a paragraph with fewer words is more likely a headline. The third and fourth rules investigate the layout and the fifth rule assigns

```
ANY{STARTSWITH(Reference) -> AuthorStart}; // Kluegl
COLON{-> AuthorEnd} CW; // : Collective
(AuthorStart # AuthorEnd){-> Author};//Kluegl ... Puppe, F.:
```

UIMA Ruta Listing 4.23: Boundary matching approach for labeling author sections. First rule detects the start position, the second rule identifies the end position, and the third rule combines both for the complete annotation.

```
Headline{-CONTAINS(W) -> UNMARK(Headline)};
```

UIMA Ruta Listing 4.24: A transformation-based rule for removing defective Headline annotations.

```
"Mr" PERIOD? @Person{-> SHIFT(Person,1,3)};
```

UIMA Ruta Listing 4.25: A transformation-based rule for expanding the offsets of an annotation.

```
STRING s;
Paragraph{CONTAINS(W,1,5)->MARKSCORE(5,HeadlineInd)};
Paragraph{CONTAINS(W,6,10)->MARKSCORE(2,HeadlineInd)};
Paragraph{CONTAINS(Bold,80,100,true)
   ->MARKSCORE(7,HeadlineInd)};
Paragraph{CONTAINS(Bold,30,80,true)
   ->MARKSCORE(3,HeadlineInd)};
Paragraph{CONTAINS(CW,50,100,true)
   ->MARKSCORE(7,HeadlineInd)};
Paragraph{CONTAINS(W,0,0)->MARKSCORE(-50,HeadlineInd)};
HeadlineInd{SCORE(10)->MARK(Headline)};
HeadlineInd{SCORE(5,10)->MATCHEDTEXT(s),
        LOG("Maybe a headline: " + s)};
```

UIMA Ruta Listing 4.26: Scoring rules for weighting different aspects of headlines. The rules create an annotation of the type Headline for paragraphs like "Diagnoses:" since the first (line 2) and fifth rule (line 6) increase the score resulting in an overall score of 12. The rule in line 8 evaluates the score and creates a new annotation.

higher scores to paragraphs with many capitalized words. The sixth rule reduces the score for paragraphs that contain no words at all. The remaining rules evaluate the score and either create a new annotation of the type Headline, if the score exceeds the threshold of 10, or emit a message for less certain positions using a string variable defined in line 1.

4.2.7 Exemplary Script

The previous sections provided examples illustrating isolated aspects of the syntax and semantic of the UIMA Ruta language. This section describes a complete script for the detection of characters in German literary texts developed by Stefan Olbrecht as part of his bachelor thesis [158]. The rules are divided in Listing 4.27 (Part 1) and Listing 4.28 (Part 2). The script is part of a small pipeline and assumes that a part-of-speech tagger and a lemmatizer have been applied before. Comments are given in German.

The script in Listing 4.27 starts with the package declaration. Then, two import statements in line 3-6 import types for part-of-speech tags using an alias and a type for lemmas from the DKPro type system [95]. These special import statements have not been discussed before in Section 4.2.2.1. They can be utilized in order to avoid problems with ambiguous short names of types. The statement in line 8 imports an additional script concerning direct speech. From line 10 to 19, various dictionaries are defined. They include gazetteers for animals, first names, phrases, jobs, objects, relatives, special verbs as well as general, military and noble titles. The lines 21-29 declare additional types and variables. From line 31 to 41, the imported script for direct speeches and the various dictionaries are applied. The dictionary matching is sensible to whitespaces specified by the rule in line 32. Then, the first typical rule is given in the lines 46-47. It create annotations of the type FirstName for a noun that is preceded by a mention of a relative or an animal. Additionally, the rule stores the covered text of the matched noun in the list FirstNames. The rule in line 50 investigates all FirstName annotations, which have been created by the last rule or by the dictionary matching, and stores their covered text extended with an additional "s" in the same list as before. Thus, the strings in the list also cover different cases. The last rule in the first part of the script joins subsequent FirstName annotations.

Part 2 of the script is given in Listing 4.28. The first rule in line 4-6 creates LastName annotations for capitalized words if they are preceded by a capitalized word that is listed in FirstNameList and by an optional word "von". Additionally, the rule stores the covered text of the potential last name in the list LastNames. The rule in line 9-12 detects last names that are preceded by any kind of title and stores the covered text in the same list. The third rule in line 15 extends the list LastNames with the covered text of each found last name extended with the string "s". The next rule in line 18-19 uses this list in order to annotate all nouns that are contained in the list with the type LastName. The two rules in line 24-29 detect mentions of characters without actual names. The first rule creates AddName annotations for animals, relatives, jobs and titles, and stores the covered text in the list AddNames. The second rule creates an AddName annotation for all nouns, whose lemmas are listed in that list. The rules in line 34-35 create PersName annotations for different patterns of AddName, FirstName and LastName annotations. The rule in line 38-40 investigates direct speeches. If a direct speech is followed by an optional comma, a special verb, whose lemma is present in the dictionary VerbList, an optional noun marker and a list of capitalized words, then these last words are annotated with the type PersName. The covered text of the created annotations is additionally

stored in the list PersNames. The rule in line 43 creates a PersName annotation for each noun, whose lemma is present in the list PersName. The remaining rules apply some final corrections. First, the rule in line 48 clears the list FirstNames. Then, the rule in line 49-51 detects first names after direct speeches and adds them to the list FirstNames. The rule of line 54-55 identifies first names after specific phrases like "I am…" and adds the covered text to the list FirstNames. The rule in line 58 investigates all nouns and annotates them with the type FirstName if they are listed in FirstNames. The last rule states that first names are also annotations of the type PersName.

Overall, this script uses different techniques in order to detect characters (PersName) in literary texts. A major part is dependent on dictionaries. The rules utilize the found keywords in sequential patterns in order to identify new annotations. Furthermore, lists are applied for finding distant mentions of the same characters.

»Nun sieh mal, Saljoshew ist auch da!« murmelte Rogoshin, indem er mit einem triumphierenden, sogar etwas boshaften Lächeln nach ihnen hinblickte; dann wandte er sich auf einmal zum Fürsten. »Fürst, ich weiß nicht, weswegen ich dich liebgewonnen habe. Vielleicht, weil ich dich in einem solchen Augenblick getroffen habe; aber den hier habe ich doch auch getroffen« (er wies auf Lebedew), »und den habe ich nicht liebgewonnen. Komm zu mir, Fürst! Wir werden dir diese Gamaschen ausziehen; ich werde dir den besten Marderpelz kaufen, dir den schönsten Frack machen lassen, eine weiße Weste oder was für eine du sonst wünschst; ich werde dir die Taschen voll Geld stopfen, und...dann wollen wir zu Nastasja Filippowna fahren! Wirst du kommen oder nicht?« »Gehen Sie darauf ein, Fürst Lew Nikolajewitsch!« fügte Lebedew in eindringlichem, feierlichem Tone hinzu. »Lassen Sie sich das ja nicht entgehen! Lassen Sie sich das ja nicht entgehen!« Fürst Myschkin stand auf, streckte Rogoshin höflich die Hand hin und sagte freundlich zu ihm:

Figure 4.3: Results of the UIMA Ruta script applied on an excerpt of Dostoevskij's "Der Idiot" (in German) with highlightings for identified mentions of characters. Missing entities are underlined.

Figure 4.3 contains an excerpt of Dostoevskij's "Der Idiot" (in German) with highlightings for the identified mentions of characters (PersName annotations). The script was only applied on this excerpt resulting in several false negative entities. They are underlined in the figure. The light grey entities can be found by adding a simple rule. If the rules are applied on the complete chapter, then all entities are identified. The first mention of "Rogoschin" and the second mention of "Lebedjew" have been created by the rule in line 38-40 due to the direct speech and the specific verb. The second mention of "Rogoschin" and the first mention of 'Lebedjew" have been created by the rule in line 43 because the previous rule added the covered text to the list PersNames. The rule in line 34 created the PersName annotations "Nastasja Filippowna" and "Fürst Ljow Nikolajewitsch" due to the presence of a FirstName annotation. "Fürst Myschkin" has been created by the rule in line 35 because of a LastName annotation, which was identified by the rule in line 9-12 using the context of the NobleTitle annotation ("Fürst"). Single AddName annotations are not annotated with PersName annotations and thus the script is not able to detect the mention of "Fürsten" and the two mentions of "Fürst". Their identification can be achieved in various ways, e.g., by adding the rule `AddName{-PARTOF(PersName) -> MARK(PersName)};` at the end of the script.

```
PACKAGE uima.ruta.firefly;

IMPORT PACKAGE de.tudarmstadt.ukp.dkpro.core.api.lexmorph.type.pos
  FROM GeneratedDKProCoreTypes AS pos;
IMPORT de.tudarmstadt.ukp.dkpro.core.api.segmentation.type.Lemma
  FROM GeneratedDKProCoreTypes;

SCRIPT uima.ruta.firefly.DirectSpeech;

WORDLIST AnimalList = 'Animals.txt';
WORDLIST FirstNameList = 'FirstNames.txt';
WORDLIST GeneralTitleList = 'GeneralTitles.txt';
WORDLIST PhraseList = 'IMPhrases.txt';
WORDLIST JobList = 'Jobs.txt';
WORDLIST MilitaryTitleList = 'MilitaryTitles.txt';
WORDLIST NobleTitleList = 'NobleTitles.txt';
WORDLIST ObjectWordList = 'ObjectWords.txt';
WORDLIST RelativesTitleList = 'RelativesTitles.txt';
WORDLIST VerbList = 'Verbs.txt';

STRINGLIST AddNames;
STRINGLIST FirstNames;
STRINGLIST LastNames;
STRINGLIST PersNames;
STRING Match;

DECLARE PersName, AddName, FirstName, LastName, AnAnimal,
RelativesTitle, GeneralTitle, MilitaryTitle, NobleTitle,
Job, BodyPart, IMPhrase;

Document{-> CALL(DirectSpeech)};
Document{-> RETAINTYPE(SPACE)};
Document{-> MARKFAST(AnAnimal, AnimalList)};
Document{-> MARKFAST(FirstName, FirstNameList)};
Document{-> MARKFAST(GeneralTitle, GeneralTitleList)};
Document{-> MARKFAST(IMPhrase, PhraseList)};
Document{-> MARKFAST(Job, JobList)};
Document{-> MARKFAST(MilitaryTitle, MilitaryTitleList)};
Document{-> MARKFAST(NobleTitle, NobleTitleList)};
Document{-> MARKFAST(RelativesTitle, RelativesTitleList)};
Document{-> RETAINTYPE};

//---- VORNAMEN -------------\\

// Titel, nach denen wahrscheinlich ein Vorname kommt, z.B. Bruder Senka
(RelativesTitle|AnAnimal) pos.N{-PARTOF(FirstName), -PARTOF(SPECIAL)
  -> MARK(FirstName), MATCHEDTEXT(Match), ADD(FirstNames, Match)};

// Vornamen, die mit "s" enden speichern
FirstName{-> MATCHEDTEXT(Match), ADD(FirstNames, Match + "s")};

// Alle Vornamen in der Liste FirstNames markieren
pos.N{-PARTOF(FirstName), INLIST(FirstNames) -> MARK(FirstName)};

// Vorname + Vorname = Vorname, z.B. Nikolai Andrejewitsch
FirstName{-> SHIFT(FirstName, 1, 2)} FirstName{-> UNMARK(FirstName)};
```

UIMA Ruta Listing 4.27: (Part 1) Script for detection mentions of characters in literary texts [158]. An explanation of the rules can be found in the corresponding section. Comments in the script are given in German.

```
//---- NACHNAMEN ------------\\

// Vornamen + CW (-> Nachname), z.B. Andrejewitsch Pawlischtschew
CW{INLIST(FirstNameList)} "von"? CW{-PARTOF(FirstName), -PARTOF(LastName),
  -PARTOF(pos.PP) -> MARK(LastName, 2, 3), MATCHEDTEXT(Match),
  ADD(LastNames, Match)};

// Titel, nach denen wahrscheinlich ein Nachname kommt, z.B. Herr Pawlisch...
(GeneralTitle|MilitaryTitle|NobleTitle|Job) "von"? CW{-PARTOF(GeneralTitle),
  -PARTOF(Job),-PARTOF(NobleTitle), -PARTOF(MilitaryTitle),
  -PARTOF(RelativesTitle), -PARTOF(FirstName), -PARTOF(LastName)
  -> MARK(LastName, 2, 3), MATCHEDTEXT(Match), ADD(LastNames, Match)};

// Nachnamen, die mit "s" enden speichern
LastName{-> MATCHEDTEXT(Match), ADD(LastNames, Match + "s")};

// Markieren der Namen in LastName
pos.N{-PARTOF(FirstName), -PARTOF(LastName), INLIST(LastNames)
  -> MARK(LastName)};

//---- ADDNAMEN -------------\\

// Verschiedene Titel zusammenfassen
(AnAnimal|RelativesTitle|GeneralTitle|MilitaryTitle|NobleTitle|Job)
  {-> MARK(AddName), MATCHEDTEXT(Match), ADD(AddNames, Match)};

// Alle Zusatztitel in der Liste AddNames markieren
pos.N{-PARTOF(AddName), INLIST(AddNames, Lemma.value)
  -> MARK(AddName)};

//---- PERSONEN -------------\\

// AddName + FirstName + LastName = PersName
(AddName* @FirstName LastName*){-PARTOF(PersName) -> MARK(PersName)};
(AddName* FirstName? @LastName){-PARTOF(PersName) -> MARK(PersName)};

// Sprecher nach direkter Rede als Person markieren
DirectSpeech COMMA? pos.V{INLIST(VerbList, Lemma.value)} pos.ART? CW+
  {-PARTOF(PersName)-> MARK(PersName), MATCHEDTEXT(Match),
  ADD(PersNames, Match)};

// Alle Personen in der Liste PersNames markieren
pos.N{-PARTOF(PersName), INLIST(PersNames, Lemma.value) -> MARK(PersName)};

//---- KORREKTUR ------------\\

// Eigennamen, die nicht als Vornamen oder Nachnamen erkannt wurden
Document{-> CLEAR(FirstNames)};
DirectSpeech COMMA? pos.V{INLIST(VerbList, Lemma.value)}
  CW+{-PARTOF(FirstName), -PARTOF(LastName), -PARTOF(AddName)
  -> MARK(FirstName), MATCHEDTEXT(Match), ADD(FirstNames, Match)};

// Eindeutige Phrasen nach denen ein Name kommt
IMPhrase CW{-PARTOF(FirstName) -> MARK(FirstName), MATCHEDTEXT(Match),
  ADD(FirstNames, Match, Match + "s")};

// Alle Vornamen in der Liste FirstNames markieren
pos.N{-PARTOF(FirstName), INLIST(FirstNames) -> MARK(FirstName)};
FirstName{-PARTOF(PersName) -> MARK(PersName)};
```

UIMA Ruta Listing 4.28: (Part 2) Script for detection mentions of characters in literary texts [158]. An explanation of the rules can be found in the corresponding section. Comments in the script are given in German.

4.3 Development Environment and Tooling

The development of rule-based applications for information extraction is first of all an engineering task. It can be a difficult, tedious and time consuming task, which depends on human resources and their qualification. Qualified rule engineers are potentially an expensive resource or only available to a limited extent. The engineers should therefore be supported by tools that ease the access to the rule-based systems and that improve the engineering experience in general. While improvements of the usability and an extensive documentation weaken the learning curve, the development environment should provide features that decrease the time and labor enabling the rapid development of rule sets. A (non-exhaustive) list of those features for supporting the rule engineer can be identified:

Clear visualization of different language elements The visualization of language elements by highlighting different syntactic constructs provides a clear overview of the rule set. Semantic highlighting of the occurrences of some language elements further improves the readability.

Direct feedback on defective rules Writing rules is an error-prone process. The development environment should notify the user instantly about syntax errors, typing error and misplaced constructs. The engineering process is decelerated if feedback about erroneous rules is given not before the rules are applied.

Shortcuts for editing The development environment should provide functionality for recurring elements either by completion of the statements of the user or by templates for favored constructs. In general, the effort of writing rules should be minimized.

Adaptable to process model of user The rule engineering task can be approached with different process models. The user should be able to adapt and optimize the development to her own process model.

Easy introspection of results When the engineered rules are applied on exemplary documents, the resulting annotations should be visualized in an clear and accessible way. The user should be able to directly investigate different aspects of the annotated documents.

Explanation of rule execution The application of rules on a document should not be opaque. The user should be able to comprehend and track every step of the rule inference in order to identify and correct undesired behavior.

Automatic validation of the rule set's correctness Manual verification of the rules and the validation of their results is tedious and time consuming and has to be performed potentially every time the rule set is extended or refined. This task should be automated by specifying test cases and informative reports should be provided to the user.

Support for automatic induction of rules The development environment should support the user in the specification of rule sets. In case additional data is available, this data can be utilized in order to propose rules that solve a specific annotation task.

The UIMA Ruta Workbench [126] tries to provide a development environment and additional tooling, which cover all of these features. It is implemented as a rich client application extending the Eclipse platform[38], which allows the user to arrange the different features and tools according

[38] `http://www.eclipse.org/`

to her preferences. The available tooling of the Eclipse platform can be directly utilized in the UIMA Ruta Workbench, e.g., version control of script files, which allows collaborative development. The large amount of features and additional tooling leads to an increased complexity of the system. Thus, the UIMA Ruta Workbench is not easily accessible for inexperienced users, but favors trained engineers, which are able to take advantage of all features for an increased productivity. The following sections highlight how a strong tooling support can render the rapid development of rule-based information extraction systems possible, and refer to figures in order to illustrate the different tooling support.

4.3.1 Basic Development Support

A development environment for rule-based information extraction applications should provide at least some basic features. These include the option to create and modify rules, the execution of these rules on a set of documents and the visualization of the annotations created by the rules. The central parts of the UIMA Ruta Workbench are the workspace with UIMA Ruta projects, the full-featured editor, the launch capacity for executing script files and the visualization of annotated documents based on the CAS Editor[39].

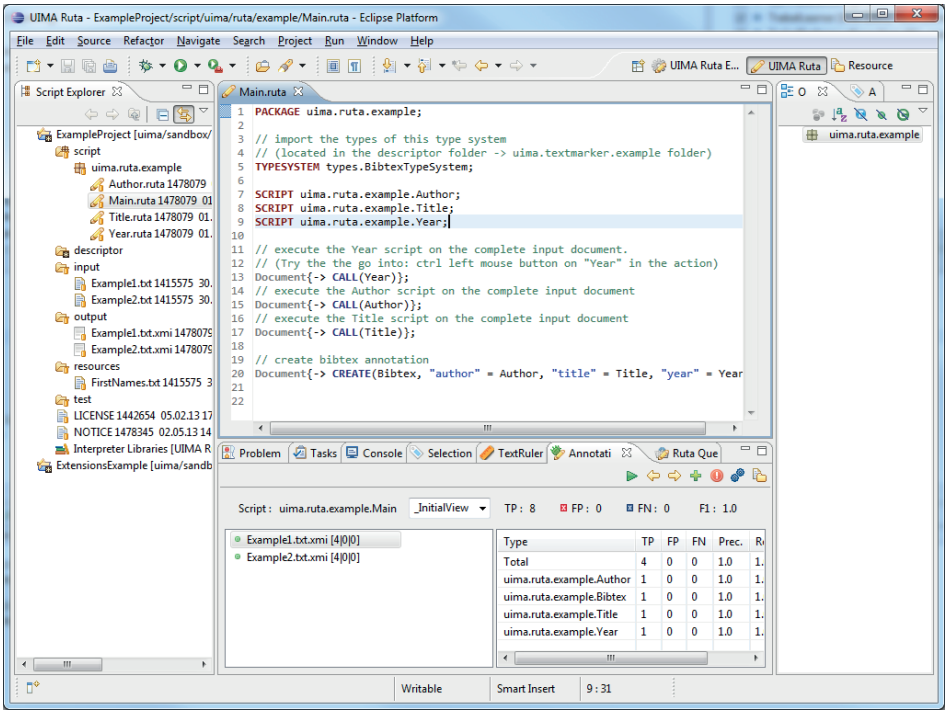


Figure 4.4: Default perspective of the UIMA Ruta Workbench: Script explorer with a UIMA Ruta project (left part). Full-featured editor for specifying rules (center part).

[39] http://uima.apache.org/d/uimaj-2.5.0//tools.html#ugr.tools.ce

Figure 4.4 provides a screenshot of the default perspective of the UIMA Ruta Workbench. The left part contains the list of UIMA Ruta projects in the workspace. A UIMA Ruta project consists of a distinct folder layout. The script folder contains the rule-based script files, the descriptor folder the descriptor files of UIMA analysis engines and type systems, and the resources folder contains dictionaries and word lists. The workbench automatically generates an analysis engine and type system descriptor for each script file. These descriptors can be utilized in order to include the UIMA Ruta rules in arbitrary UIMA applications. By default, when executing a script file, documents in the input folder are processed and their results are stored in the output folder. The UIMA Ruta Workbench supports mixin-workspaces. The user is able to add dependencies in a UIMA Ruta project pointing to other projects of the workspace. A dependency to a UIMA Ruta project enables the user to refer to its script files and leads to reusable subprojects. The Workbench takes care of the correct configuration of the generated analysis engines descriptors. A UIMA Ruta project can, however, also have dependencies to Java projects in the workspace. When a UIMA Ruta script is launched, then the classpath is automatically expanded by the dependencies and analysis engines implemented in the same workspace can be utilized in a script file. This enables rapid prototyping across languages.

The central part of the screenshot shows the full-featured editor, which provides the editing support known by common development environments. These include syntax highlighting, semantic highlighting, syntax checking, auto-completion, template-based completion and others. Thereby, the user is optimally supported when writing or editing rules. Defective rules are instantly highlighted in the editor and the auto-completion proposes suitable language elements based on the current editing position.

For visualizing the results of the rule execution, the UIMA Ruta Workbench utilized the CAS Editor extended with additional views for an improved access to the annotations. The *Annotation Browser* view lists the annotations of a document sorted by their types. Figure 4.9 contains a screenshot of this view. The *Selection* view lists the annotations of the currently selected position in the CAS Editor sorted by their types (cf. Figure 4.5). Both views provide additional filtering options and enable the user to obtain a fast overview of the results of the rule execution and to investigate specific positions.

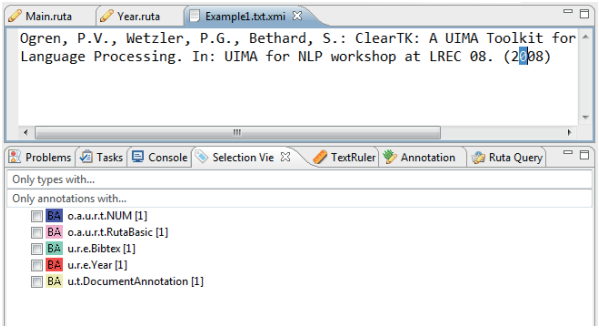


Figure 4.5: Selection view for listing annotation at the selected position

4.3.2 Explanation of Rule Execution

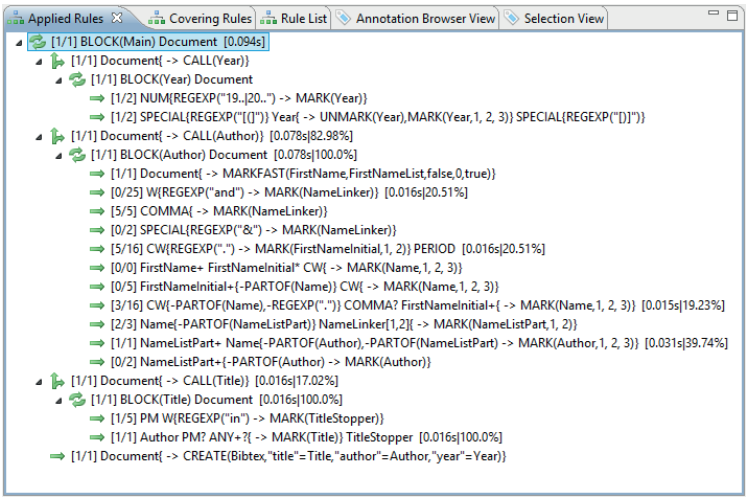


Figure 4.6: Applied Rules view: Listing of the applied rules with information of the amount of successful and failed executions, and about the absolute and relative execution time.

The main view for the explanation is the *Applied Rules view*. It displays structured information about all rules that tried to apply to the document (cf. Figure 4.6). The structure is as follows: if BLOCK constructs were used in the executed UIMA Ruta script, then the rules contained in that block will be represented as child nodes in the tree of the view. Each Ruta file is a BLOCK construct itself and named after the file. The root node of the view is, therefore, always a BLOCK containing the rules of the executed UIMA Ruta script. Additionally, if a rule calls a different Ruta file, then the root block of that file is the child of the calling rule. Besides the hierarchy, the view shows how often a rule tried to match in total and how often it succeeded. This information is given in brackets at the beginning of each rule entry. The rule itself is given afterwards. Additionally, some profiling information, giving details about the absolute execution time and the relative execution time within the current block is added at the end of each rule entry.

The *Matched* and *Failed* views display the rule matches the rule selected in the *Applied Rules* view (cf. Figure 4.7). The views contain the covered texts that provide a quick overview to the user, which positions have been successfully matched and which positions have been unsuccessfully investigated by the rules.

The *Rule Elements* views finally displays the exact covered text for each match reference and the evaluation results for each condition. This enables the user to quickly identify the causes for a successful or failed rule match.

The views so far for the explanation of the rule execution present a well-structured report if the user tries to inspect the behavior of specific rules. If, however, several rules are creating a certain type of annotation, then the user has to investigate each of these rules in order to identify the reasons for an erroneous annotation. The *Created By* view solves this deficiency by

Figure 4.7: The Matched and Failed views display the rule matches the rule selected in the Applied Rules view.

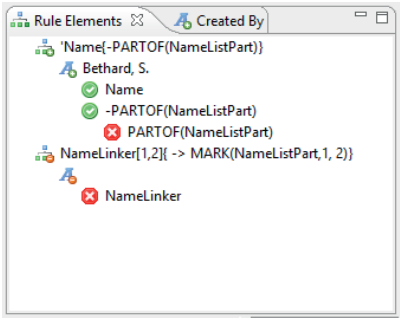


Figure 4.8: The Rule Elements view provides a complete listing of the matched positions of the match references and the results of the evaluated conditions.

displaying the rule that was responsible for the creation of a specific annotation that has been selected in the *Annotation Browser* view (cf. Figure 4.9).

The UIMA Ruta Workbench provides additionally several other views that help the user to investigate the rule execution in an efficient manner. To these belong views that visualize the rule matches on specific positions or the rule matches of rules containing specific language elements, as well as the *Statistics* view, which provides a compact report of the execution time of the conditions and actions.

4.3.3 Introspection by Querying

The engineered rules are not only applied on a single document during the development process, but on a collection of documents. In order to investigate the resulting annotations, the user typically has to inspect each document separately. The UIMA Ruta Workbench provides an additional view for the introspection in collections of annotated documents by querying. The *Ruta Query* view contains different text fields for specifying the folder containing the queried

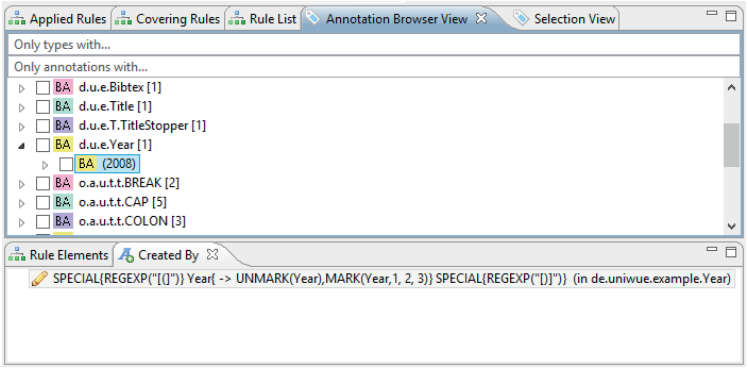


Figure 4.9: Created By view displaying the rule, which created the annotation selected the Annotation Browser view.

documents, the necessary type system and a set of UIMA Ruta rules (cf. Figure 4.10). The view applies the rules on the collection of documents and presents the matches of the rules to the user. The rules can thus be interpreted as a query statement. The view can be applied in many scenarios during the development process. The user can, for example, obtain an overview of the occurrences of certain types of annotations or even patterns of annotations. This includes useful activities like investigating if a pattern of annotations has been or has not been annotated with a specific type. In contrast to work like [55] and [89] on querying and retrieval, this view is only intended to support the knowledge engineer since no indexing is applied and the query does not scale for other applications.

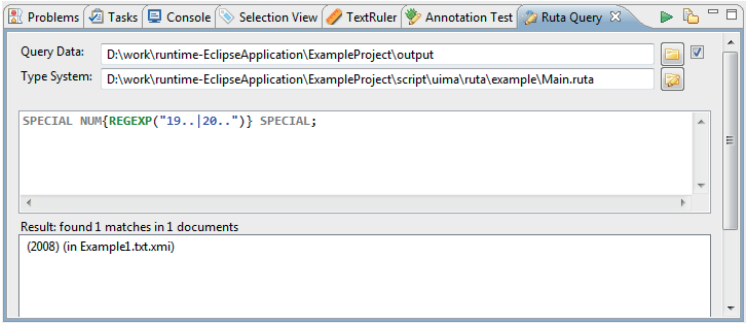


Figure 4.10: The Ruta Query view displays the rule matches in a set of documents.

4.3.4 Automatic Validation

The manual validation of the results of a currently developed rule set can be tedious and time consuming, especially if several documents have to be inspected. The user has to perform this

task potentially each time the rule set is extended or refined in order to ensure the correct processing of the documents. An automatic validation of the rule set's correctness is therefore one of the most important features of a development environment for rule-based information extraction. The user should be able to define test cases that are utilized to validate the rule set. In the context of information extraction applications, the test cases are equivalent to a set of documents containing annotations of interesting types. The rules are applied on the raw documents and the resulting annotations created by the rules are then compared to the given gold annotations. The differences are used to calculate an evaluation score, e.g., the F_1 score[40].

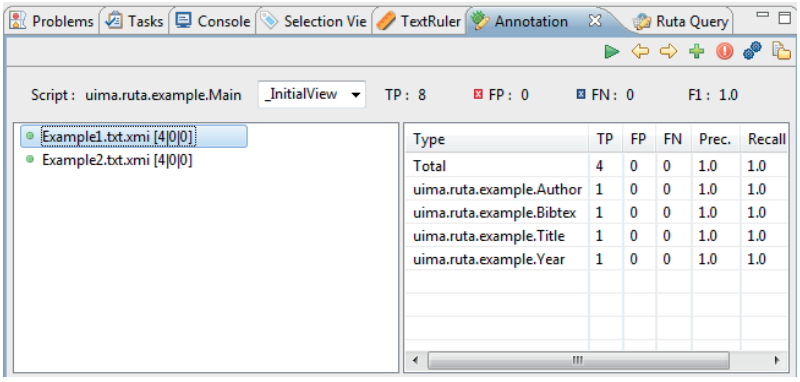


Figure 4.11: The Annotation Testing view for automatic validation of rule sets using annotated documents.

The UIMA Ruta Workbench provides this functionality with the *Annotation Testing* view. Figure 4.11 contains a screenshot of this view, which consists of the list of tested documents on the left side and the results for the currently selected document on the right side. Additionally the user is able to select different evaluators resulting in different F_1 scores, e.g., based on annotations or tokens. The usefulness of this functionality can be summarized with the description of three use cases:

Goal-oriented development A real-world scenario for the development of a rule-based information extraction system often includes a quality threshold specified by a contractee (cf. [187]). Given a set of annotated documents, the developed rules have to achieve a previously defined evaluation score in order to be accepted for deployment. This use case is directly supported by the UIMA Ruta Workbench and the user is able to continuously compare the current state of the system to the desired targets.

Test-driven development Test-driven development [107] is a programming process model where first test cases for specific parts of functionality are defined before the actual program code is written. The developed code is then continuously validated during development in

[40] Parts of the contents of this section have been published in Peter Kluegl, Martin Atzmueller, and Frank Puppe. Test-driven Development of Complex Information Extraction Systems using TextMarker. In Grzegorz J. Naplepa and Joachim Baumeister, editors, 4th International Workshop on Knowledge Engineering and Software Engineering (KESE 2008), 31th German Conference on Artificial Intelligence (KI-2008), pages 19–30, 2008. [120]

order to ensure the correctness of different parts of the software. Test-driven development has proven itself as an effective process model [148]. This process model can also be applied to the development of rule-based information extraction applications. First, a set of documents is manually annotated, which provides a best possible coverage of different challenges. The rules are then developed against these test cases until the performance of the rules is sufficient. A methodology and detailed process model for test-driven development of rule-based information extraction systems has been published by Kluegl et al. [120].

Regression testing Another reliable process model for developing rules can be summarized as follows. The rule engineer considers one document after each other and creates new rules or refines existing rules. Starting with the first document, she creates an initial set of rules, which extracts all interesting information within this document. After the rules are applied on the document, it is either perfectly annotated or manually corrected, and is stored as a test case. The rule engineer continues with the second document and extends, modifies and refactors the rule set until the annotations in the second document are correctly identified. During these modifications of the rules set, however, the test case of the first document is continuously validated in order to ensure that the rules still provide the necessary functionality for the first document. This procedure is iterated for the complete collection of documents until all documents are sufficiently processed by the created rules.

4.3.5 Constraint-driven Evaluation

One of the advantages of rule-based information extraction approaches is that annotated documents are not strictly necessary for application development. Nevertheless, they are very beneficial for rule-based systems, e.g., for accelerating the development or quality maintenance. There is a natural lack of labeled data in most application domains and its creation is error-prone, cumbersome and time-consuming as is the manual validation of the extraction results by a human. A human is able to validate the created annotations in those documents using background knowledge and expectations on the domain. An automatic estimation of the rule set's quality in unseen or unlabeled documents using this kind of knowledge provides many advantages and greatly improves the engineering experience[41].

The Constraint-driven Evaluation (CDE) framework [203] is a combination of the testing framework and the querying functionality, and greatly improves the engineering process. It allows the user to specify expectations about the domain in form of constraints. These constraints are applied on documents with annotations, which have been created by a set of rules. The results of the constraints are aggregated to a single CDE score, which reflects how well the created annotations fulfill the user's expectations and thus provide a predicted measurement of the rule set's quality for these unlabeled documents. The documents can be ranked according to the CDE score, which provides an intelligent report about the well and poorly processed examples.

[41] The contents of this section have been published in Andreas Wittek, Martin Toepfer, Georg Fette, Peter Klügl, and Frank Puppe. Constraint-driven Evaluation in UIMA Ruta. In Peter Klügl, Richard Eckart de Castilho, and Katrin Tomanek, editors, UIMA@GSCL, volume 1038 of CEUR Workshop Proceedings, pages 58–65. CEUR-WS.org, 2013. [203]

Figure 4.12 provides a screenshot of the CDE perspective, which includes different views to formalize the set of constraints and to present the predicted quality of the model for the specified documents.

Compared to the test-driven development in the last section, this approach facilitates constraint-driven development, which requires no annotated data. This process is illustrated with a simple example for identifying one specific person name in each document of a larger collection. The knowledge engineer specifies her background knowledge and expectations about the domain using rules. These rules cover, for example, that each document should contain exactly one annotation of the type Name. After the actual extraction rules are applied on the large set of documents, the expectations are compared to the Name annotations created by the rules. Using a score called CDE score, the documents are ranked whereas documents that violate the expectations are listed first. The knowledge engineer is now able to investigate these problematic documents where the rule obviously failed either by finding no name or by labeling multiple names. After the rule set is refined or extended, this process is iterated.

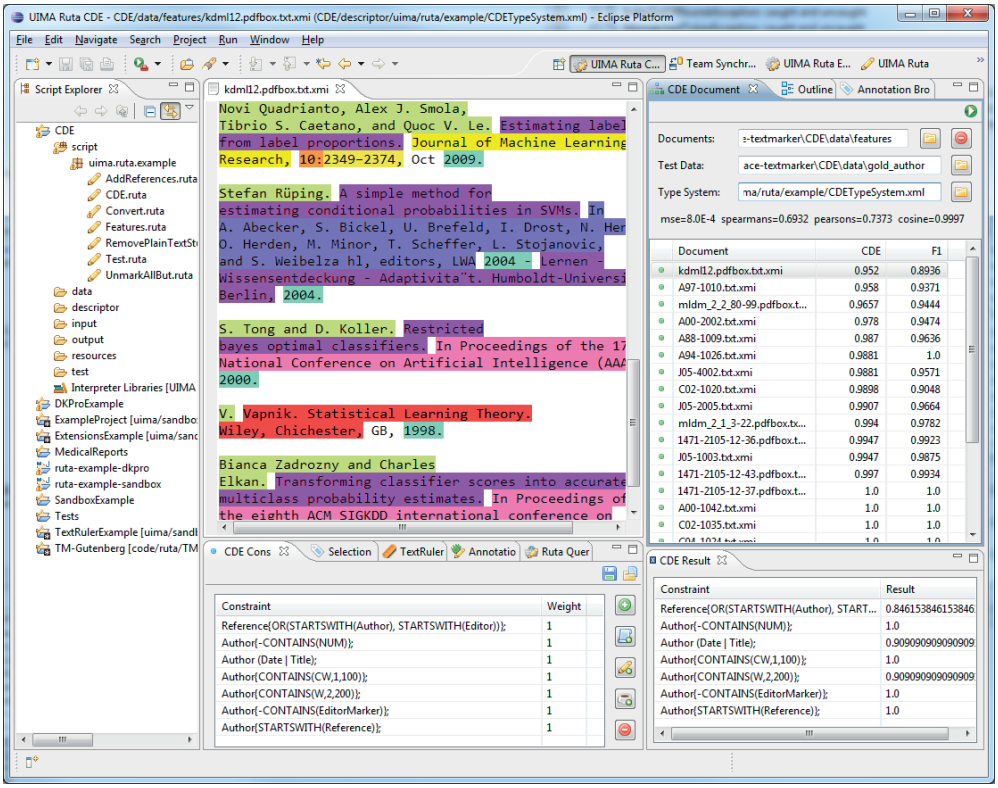


Figure 4.12: CDE perspective in the UIMA Ruta Workbench. Bottom left: Expectations on the domain formalized as constraints. Top right: Set of documents and their cde scores. Bottom right: Results of the constraints for selected document. Overall, three constraints are violated to a certain extend, e.g., an Author annotation was detected that contains less than two words (5th constraint, 4th reference).

A constraint is defined as a function $C : \text{CAS} \rightarrow [0,1]$, which returns a confidence value for an annotated document where high values indicate that the expectations are fulfilled. Two different types of constraints are supported: RULE constraints are simple UIMA Ruta rules without actions and allow to specify sequential patterns or other relationships between annotations that need to be fulfilled. The result is basically the ratio of how often the rule has tried to match compared to how often the rule has actually matched. An example for such a constraint is `Document{CONTAINS(Author)};`, which specifies that each document must contain an annotation of the type *Author*. The second type of supported constraints are Annotation Distribution (AD) constraints (c.f. Generalized Expectations [146]). Here, the expected distribution of an annotation or word is given for the evaluated types. The result of the constraint is the cosine similarity of the expected and the observed presence of the annotation or word within annotations of the given types. A constraint like `"Peter": Author 0.9, Title 0.1`, for example, indicates that the word "Peter" should rather be covered by an *Author* annotation than by a *Title* annotation. The set of constraints and their weights can be defined using the *CDE Constraint* view (c.f. Figure 4.12, bottom left).

For a given set of constraints $C = \{C_1, C_2 ... C_n\}$ and corresponding weights $w = \{w_1, w_2, ..., w_n\}$, the CDE score for each document is defined by the weighted average:

$$\text{CDE} = \frac{1}{\sum_i^n w_i} \sum_i^n w_i \cdot C_i \tag{4.1}$$

The CDE scores for a set of documents may already be very useful as a report how well the annotations comply with the expectations on the domain. However, one can further distinguish two tasks for CDE: the prediction of the actual evaluation score of the rules, e.g., the token-based F_1 score, and the prediction of the quality ranking of the documents. While the former task can give answers how good the rules perform or whether the rules are already good enough for the application, the latter task provides a useful tool for introspection: Which documents are poorly labeled by the rules? Where should the set of rules be improved? Are the expectations on the domain realistic? The CDE scores for the annotated documents are depicted in the *CDE Documents* view (c.f. Figure 4.12, top right). The result of each constraint for the currently selected document is given in the *CDE Results* view (c.f. Figure 4.12, bottom right).

The development of the constraints needs to be supported by tooling in order to achieve an improved prediction in the intended task. If the user extends or refines the expectations on the domain, then a feedback whether the prediction has improved or deteriorated is very valuable. For this purpose, the framework provides functionality to evaluate the prediction quality of the constraints itself. Given a set of documents with gold annotations, the CDE score of each document can be compared to the actual F_1 score. Four measures are applied to evaluate the prediction quality of the constraints: the mean squared error, the Spearman's rank correlation coefficient, the Pearson correlation coefficient and the cosine similarity. For optimizing the constraints to approximate the actual F_1 score, the Pearson's r is maximized, and for improving the predicted ranking, the Spearman's ρ is maximized. If documents with gold annotations are available, then the F_1 scores and the values of the four evaluation measures are given in the *CDE Documents* view (c.f. Figure 4.12, top right).

4.3.6 Supervised Rule Induction

All development support described until now has focused on the manual engineering of rule-based information extraction systems. The user defines the set of rules herself in an ecosystem of tools, which facilitate the writing or enable an improved quality maintenance. Besides that, the development environment can also support the user in the construction of new rules. Given a set of annotated documents, machine learning algorithms can be applied in a supervised fashion in order to propose new rules. The user can then inspect the proposed rules for insights in possibly interesting patterns of annotations. Furthermore, she can extend her rule set with a selection of the proposed rules, which can again be extended or adapted[42].

The UIMA Ruta Workbench provides the *TextRuler* framework for the supervised induction of rules. Figure 4.13 depicts the TextRuler view, which allows the user to specify the training data, the interesting types of annotations and the preferred learning algorithm. The induced rules for each algorithm are presented in a separate view. Currently, four different algorithms are available.

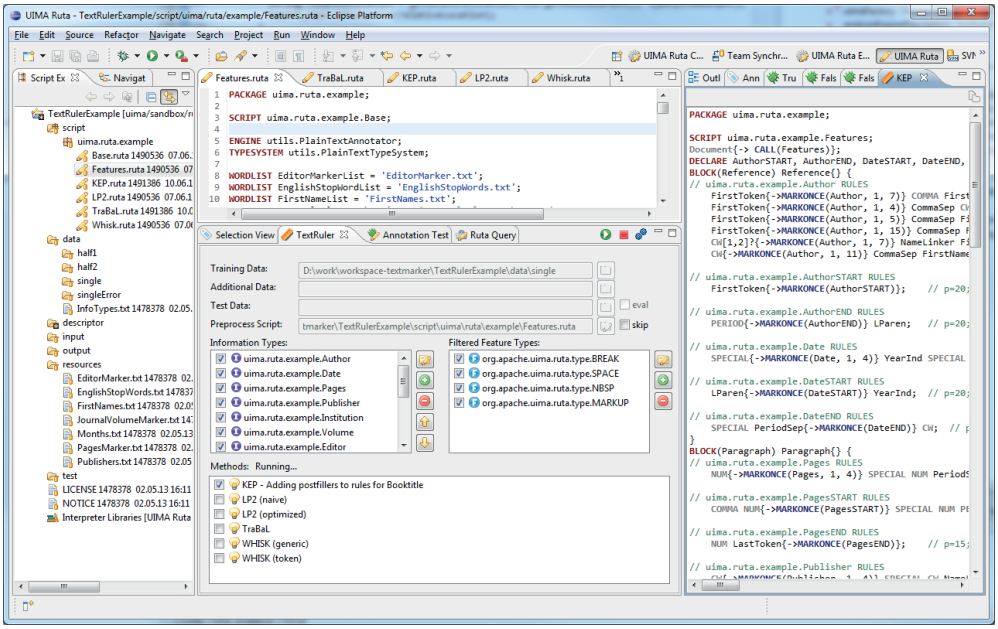


Figure 4.13: The TextRuler framework in the UIMA Ruta Workbench for supervised rule induction: configuration of the training data and the rule learning algorithms (bottom part) and the induced rules of the algorithms (right part).

[42] Parts of the contents of this section have been published in Peter Kluegl, Martin Atzmueller, Tobias Hermann, and Frank Puppe. A Framework for Semi-Automatic Development of Rule-based Information Extraction Applications. In Melanie Hartmann und Frederik Janssen, editor, Proc. LWA 2009 (KDML - Special Track on Knowledge Discovery and Machine Learning), pages 56–59, 2009. [118]

LP^2 The two implementations of this rule learner, naive and optimized, are adaptations of the original algorithm published by Ciravegna [45] for the UIMA Ruta language. LP^2 induces three different kinds of rules. Tagging rules identify the boundaries of the annotations, context rules shift misplaced boundaries and correction rules finally are able to remove boundaries again (cf. Section 2.2.3.3). Correction rules are, however, not yet supported by the implementations.

WHISK The two implementations of this rule learner, token and generic, are adaptations of the algorithm published by Soderland et al. [186] for the UIMA Ruta language. The Whisk algorithm induces rules in the form of modified regular expressions (cf. Section 2.2.3.6). In contrast to the original algorithm, the implementations do not directly support multi-slot rules.

KEP The name of the rule learner KEP (knowledge engineering patterns) [32] is derived from the idea that humans use different engineering patterns to write annotation rules. The algorithm implements simple rule induction methods for some patterns, such as boundary detection or annotation-based restriction of the window. The results are then combined in order to take advantage of the interaction of the different kinds of induced rules. Since the single rules are constructed according to how humans engineer the annotations rules, the resulting rule set resembles more handcrafted rule sets. Furthermore, by exploiting the synergy of the patterns, the patterns for some annotations are much simpler.

TraBaL The TraBaL algorithm [69] is able to induce transformation-based error-driven rules. The basic idea is similar to the Brill-Tagger [34], but template generation is more generic and can also handle arbitrary annotations instead of tags of tokens. This algorithm was built in order to learn how to correct the annotations of arbitrary models or human annotators.

A methodology and process model for the semi-automatic development of rule-based information extraction systems where the human engineer is supported and completed by rule induction algorithms has been published in Kluegl et al. [118].

4.3.7 Semi-automatic Creation of Gold Documents

The last sections highlighted the importance of annotated documents for the development of rule-based information extractions system. A popular approach for creating annotated documents is the semi-automatic annotation using rules. The labeling of a large collection of documents is time consuming, but can be accelerated if recurring annotations are automatically created. The rule engineer defines a few rules that are able to create correct annotations instead of manually specifying them one after each other. Afterwards, the created annotations need to be verified, whereas missing annotations are added and defective annotations are removed. This functionality is already covered by the default tooling of the UIMA Workbench in combination with the CAS Editor. The approach can, however, be improved if the user is supported in the manual verification of the annotations. The UIMA Ruta Workbench provides the additional view *Check Annotations* for this use case, which enables the user to efficiently accept, reject or replace the proposed annotations. Figure 4.14 contains a screenshot of this view located in the

right part. The view lists all annotations of types selected by the user, which can quickly classify these annotations as correct or erroneous.

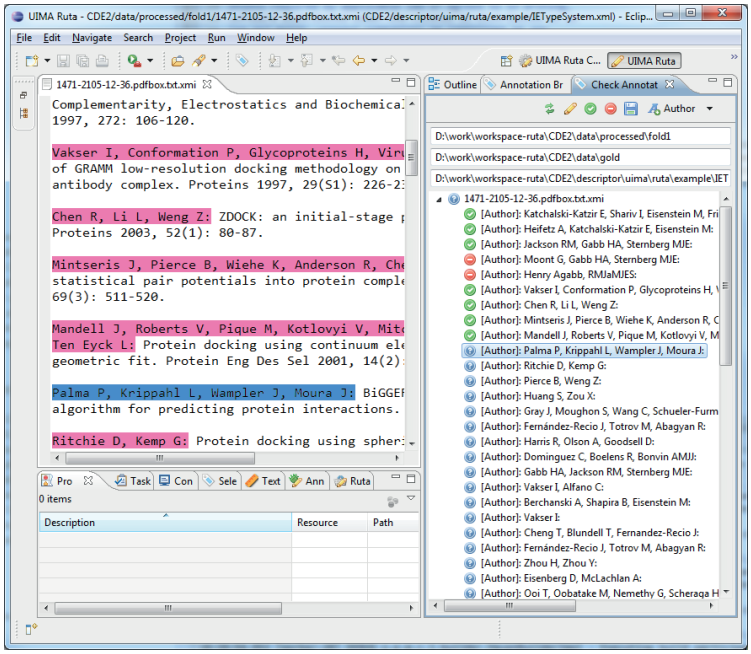


Figure 4.14: The Check Annotations view for efficiently verifying the correctness of annotations in a collection of documents.

4.4 Comparison to Related Systems

UIMA Ruta is compared to a representative selection of related systems and highlights different aspects of rule representation and execution, expressiveness of the language, runtime performance, and available tooling for development support. A special focus is laid on the concise representation of rules, which is one important aspect for rapid development. The less text the knowledge engineer has to write for achieving the same functionality, the better. The compactness and expressiveness of the UIMA Ruta language is illustrated in Figures 4.15, 4.17 and 4.18. Each figure depicts a representative example of a related language taken from the respective publication and its equivalent in the UIMA Ruta language.

```
Macro: AMOUNT_NUMBER
({Token.kind == number}
 (({Token.string == ","} |
   {Token.string == "."})
  {Token.kind == number})*
)
Rule: Money1
(
  (AMOUNT_NUMBER)
  (SpaceToken.kind == space)?
  ({Lookup.majorType == currency_unit})
)
:money  -->
  :money.Number = {kind = "money", rule = "Money1"}
```

```
(NUM ((",") | (".") NUM)*)
    {-> AmountNumber};
(AmountNumber SPACE? CurrencyUnit)
    {-> Money};
```

Figure 4.15: An excerpt of an exemplary JAPE macro and rule [54] (left) for the detection of "money" entities and their UIMA Ruta equivalents (right).

Figure 4.15 contains an example of a JAPE macro and rule [54] and their equivalents in UIMA Ruta. This example assumes that whitespaces are not filtered, that the type "Money" inherits from the type "Number", and that gazetteers directly create annotations of the specified types, as it is usual in UIMA Ruta. UIMA Ruta does not enforce a clear separation of conditions and actions and thus does not need to support labels. Java code cannot directly be included in UIMA Ruta rules, but the language itself can be extended, and arbitrary Java code wrapped in additional analysis engines can be executed. JAPE specifies the accessible types for each phase, whereas UIMA Ruta applies a more complex and dynamic paradigm of visibility controlled by the annotations themselves. In contrast to JAPE, which compiles all rules of a phase into one FST, UIMA Ruta applies the rules sequentially in the order they are specified, supports variable matching direction, and is able to match on all disjunctive alternatives. Overall, the UIMA Ruta language provides almost all features of JAPE together with a more concise representation. The development of JAPE grammars is barely supported by tooling to the best knowledge of the authors. The GATE framework provides, however, a rich selection of tools like ANNIC [55] for discovering new patterns.

For a comparison of the runtime performance, the Named Entity Recognition component in GATE is utilized without the part-of-speech tagger and applied only the first three phases of the JAPE grammar with overall 58 rules. A translation of these rules to UIMA Ruta resulted in about 44 rules, which can be considered as created by an inexperienced knowledge engineer. They do not include UIMA Ruta specific optimizations. Both systems are applied on nine sets of documents containing each 10000 documents and providing an increasing document size, taken from the Enron email dataset[43]. Startup time of the frameworks and Java can be neglected, because the first batch was evaluated twice, but only the second run was measured. Figure 4.16 depicts the average runtime for components based on the JAPE implementation, the optimized JAPE Plus implementation[44], UIMA Ruta and UIMA Ruta with activated dynamic anchoring[45]. UIMA Ruta is able to compete well for all sizes of documents, although the rules are not optimized at all, and they apply sequentially 44 phases in contrast to three phases of

[43] https://www.cs.cmu.edu/~enron/
[44] http://gate.ac.uk/sale/tao/splitch8.html#sec:jape:plus
[45] GATE 7.1 and UIMA Ruta 2.2.0 are applied.

JAPE. Dynamic anchoring improves the performance only slightly since the patterns have not been engineered accordingly and considerable processing time is caused by the gazetteers.

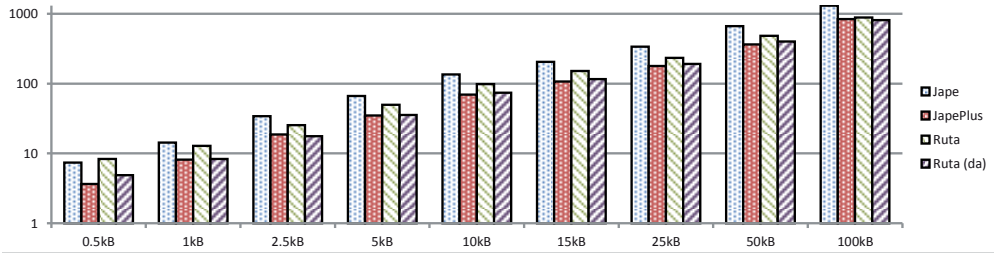


Figure 4.16: Average processing time for documents of different sizes.

The UIMA Ruta language provides a higher expressiveness than AFST [26], which is limited to a linear path through the annotation lattices. Each special functionality in AFST (honour, focus, advance, …) is available in UIMA Ruta using the corresponding language constructs. Figure 4.17 contains exemplary rules for vertical navigation in AFST and UIMA Ruta. The AFST rule starts by matching on a PName annotation and then steps into this annotation indicated by the operator "@descend". The next element "Title[string=="General"]" specifies that the PName annotation has to start with a Title annotation with the covered text "General". A period between two elements indicates a sequential constraint. The vertical navigation is repeated for the Name annotation, which has to contain a Last annotation with the covered text "Grant". The elements First and Middle are optional, but required for the linear path through the annotation lattices.

AFST includes a small set of additional predicates, whereas UIMA Ruta ships an extensive set of conditions and actions. The Domain Adaptation Toolkit [27] provides grammar development functionality and is able to create type system descriptors based on the grammars like the UIMA Ruta Workbench.

```
findG = PName[@descend] .
          Title[string=="General"] .
          Name[@descend] .
            First[]|<E> . Middle[]|<E>
            . Last[string=="Grant"] .
          Name[@ascend] .
        PName[@ascend] ;
```

```
PName<-{
  Title{REGEXP("General")}
  Name<-{
    Last{REGEXP("Grant")};
  };
};
```

Figure 4.17: An exemplary AFST rule [26] (left) for vertical matching in "PName" annotations and its UIMA Ruta equivalent (right). The rules match on text passages like "General Ulysses S. Grant" if the corresponding annotations are present. The optional patterns for the First and Middle annotations are not necessary in UIMA Ruta.

SystemT criticizes three aspects of rule languages based on the CPSL specification: Lossy sequencing, rigid matching priority and limited expressiveness in rule patterns [43]. None of these properties can be observed in UIMA Ruta. The rule language of SystemT, AQL, is a declarative relational language similar to SQL and thus does not provide a compact representation. Especially since the modification of the content of a view enforces the specification of

another view. While the syntax is accessible to programmers, it might appear counterintuitive for users not familiar with SQL. Most of the features of AQL are also supported in the UIMA Ruta language. Figure 4.18 provides an excerpt of an AQL grammar for the detection of persons and the UIMA Ruta rules with the same functionality. Broadly speaking, AQL rules typically consist of a `create view` statement that specifies the created type of annotation, `select` and `from` statement for specifying the input and output, and a `where` statement for the pattern. The first UIMA Ruta rule is equivalent to the first `create view` statement in the AQL example. There is no need for the second `create view` statement in UIMA Ruta since the rules are able to operate on the same types of annotations. The second and third UIMA Ruta rules emulate the last `create view` statement, which consolidates overlapping annotations.

```
create view CapsLast as
select  CombineSpans(C.name, L.name) as name
from    Caps C, Last L
where   FollowTok(C.name, L.name, 0 0);
...
create view PersonAll as
  (select R.name from FirstLast R) union all ...
  ... union all (select R.name from CapsLast R);

create view Person as select * from PersonAll R
consolidate on R.name using 'ContainedWithin';
```

```
(Caps Last){-> Person};
Person{PARTOFNEQ(Person)
  -> UNMARK(Person)};
Person{CONTAINS(Person,2,100)
  -> UNMARK(Person)}
```

Figure 4.18: Excerpt of exemplary AQL rules [43] (left) for the detection of persons and their UIMA Ruta equivalents (right). The last two UIMA Ruta rules are only necessary for the consolidate statement.

The rule execution of SystemT is not based on finite-state transducers, but applies an optimized operator plan for the execution of rules. The rules in UIMA Ruta are also not limited to a left-to-right matching, which can greatly improve the runtime performance. The automatic selection of the starting rule element (dynamic anchoring) is a first step towards an optimized execution plan. UIMA Ruta was developed for the rapid development of rules and cannot (yet) compete with SystemT concerning runtime performance.

SystemT provides the best tools for development support of all related systems, to the best knowledge of the author. Most of the features are also supported in a similar form by the UIMA Ruta Workbench. The development support of UIMA Ruta provides more possibilities to automatically estimate the quality of the rules, e.g., also on unlabeled documents, which is an essential assistance for developing rules. Another development environment for creating rules is the IBM Content Analytics Studio, which propagates a drag-and-drop paradigm for specifying patterns instead of a textual language. In contrast to UIMA Ruta, the system provides a more sophisticated dictionary support, but lacks many advantages of flexible rule languages. In the experience of th author, trained knowledge engineers are faster in specifying rule sets in a textual form.

UIMA Ruta is a useful tool for rule-based information extraction in the ecosystem of UIMA. The system was designed with a special focus on rapid development in order to reduce development time and costs. The rule language can be applied for solving a great number of various use cases, but still provides a compact representation. It covers most functionality of related languages and still introduces a few new features that ease the specification of complex patterns.

The UIMA Ruta Workbench adds another important aspect for rendering rapid development possible. It provides an extensive tooling support for all tasks that a knowledge engineer has to perform when creating rule-based information extraction applications. UIMA Ruta is currently unique concerning the combination of integration in UIMA, expressiveness of its language and industry-friendly open source license. The system was already applied for developing various information extraction applications. Among others, these include structured data acquisition in medical reports and resumes [9], and authoring of eLeanring systems [99].

Chapter 5

Knowledge Engineering Approaches

The last section introduced the UIMA Ruta system that provides many advantages over similar systems. Besides others features, it helps to limit the development costs of rule-based information extraction applications by providing a compact and powerful rule language together with strong tooling support. The system by itself is applicable for the creation of arbitrary natural language applications. One important factor in the development of UIMA Ruta was, however, the efficient utilization of context-specific consistencies in the handcrafted rule sets. The language provides various elements that enable the specification of consistencies and their generic usage in rules for context-specific applications. Examples for these elements cover variables, lists, specific actions, and operations on feature structures. A variable for a type of an annotation can be set to a specific value while processing a document. Rules that use these variables are automatically adapted to the circumstances in the current document and operate context-specific. Another example are lists, which can be utilized to gain knowledge about the dominant composition of entities in the current document, or actions that allow efficient modification of inconsistent entities. While some of these features can also be implemented with other rule-based systems, hardly any of them support all characteristics that allow a rapid development of applications exploiting context-specific consistencies.

This chapter provides an overview of rule-based applications developed with UIMA Ruta that take advantage of the context-specific consistencies. Some case studies have been created with the predecessor system named TextMarker. The case studies cover information extraction tasks in the three different domains introduced in Chapter 3. They apply different approaches for exploiting the consistencies, which is reflected by the structure of this chapter. The first case study describes precision-driven prototypes developed with minimal labor. Nevertheless, these rule sets are able to achieve notable results because the meager recall is gained by identifying additional entities based on the present consistencies. The second case study investigates a more sophisticated approach that introduces a separate phase for modeling the consistencies extended by an additional step that applies transformation-based rules. These rules operate on the manifestation of the consistency model by matching consistent and conflicting positions. By using transformation-based rules, the entities identified by previous steps do not need to be created again, but only corrected. The third case study incorporates ideas of the initial prototypes in a complete real-world application and highlights the usefulness of the consistencies in a thorough engineered rule set.

The applied rule sets in each case study are described mostly conceptually in order to explain the overall approach. Additional examples of rules for specific parts of the application provide insights in the actual implementation. More examples and complete rule sets can be found in the corresponding publications.

5.1 Improving Recall in Precision-driven Prototyping

This section gives an overview on two case studies that investigate the advantages of consistency-based rules for increasing the recall in precision-driven approaches. The implemented rules for exploiting the context-specific consistencies can be considered prototypes for solving a limited information extraction task in an efficient manner and with minimal engineering overhead. The targeted domains concern information extraction in curricula vitae and identification of headlines in clinical discharge letters. Both domains have already been introduced in Chapter 3 in detail, but are shortly recapitulated for convenience[46].

The limited task of information extraction in curricula vitae deals with the identification of specific entities in the past work experience of the author. In each project description of the curricula vitae, the company, sector or contractor is extracted. The company can often be identified using simple features, e.g., common suffixes, lists of known organizations or locations. Yet, these word lists cannot be exhaustive, and are often limited for efficiency reasons, e.g., for different countries. This can reduce the accuracy of the model if the employee had been working in another country for some time. Humans solve these problems of missing features, for example unknown company names, by transferring patterns of already extracted information. If the company, for example, was found in the third line of ninety percent of all project sections, then it is highly probable that the missing company can be found in the same position.

Medical discharge letters contain, for example, the examinations, the laboratory data, and the diagnoses of the patient leaving the hospital. The information extraction task needs to identify the headlines for further segmentation of the letter and identification of the performed examinations. Since there are no restrictions for writing these documents, the authors apply a variety of layouts for structuring the headlines. Humans are able to identify common headlines using the contained words. Then, they transfer these patterns to other text fragments and extract headlines with a similar layout.

5.1.1 Rule Sets

In both domains, a limited set of rules is already able to produce noteworthy results by exploiting these context-specific consistencies. Figure 5.1 provides an overview of a generic process model. The manifestation of the process model as a rule set is addressed in the following. First, common features for identifying the interesting information are created. These cover dictionaries for specific keywords, prefixes or locations, and annotations for layout information and relevant patterns. A static information extraction model identifies initial entities with minimal effort. Rules implementing expectations can help to identify highly confident information and relevant meta-features representing the properties of the consistencies. Examples for these meta-features are the distinct line where the companies occur, or the distinct font of headlines.

In precision-driven rule sets, the identified information is already highly confident so that only the properties need to be modelled. Transfer knowledge is responsible for the projection or comparison of the given meta-feature in order to identify missing entities and thus increasing the recall of the rule set. This process can be iteratively applied for including an incremental set of meta-features.

[46] The content of this section has been published in Peter Kluegl, Martin Atzmueller, Frank Puppe. Meta-level Information Extraction. KI, volume 5803 of Lecture Notes in Computer Science, Springer, 2009. [121]

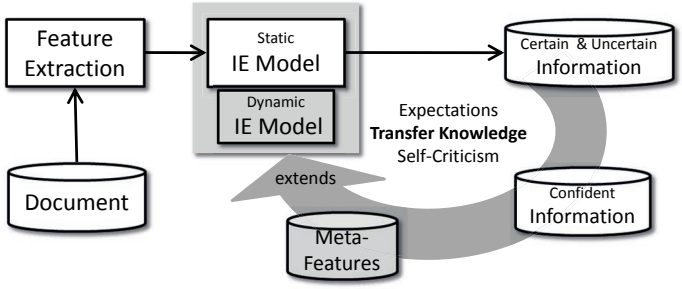


Figure 5.1: Process model with meta-features and transfer knowledge [121].

The elements of the generic process model are described in [121]:

Meta-Features Relations between features and information, respectively patterns, are explicitly implemented by meta-features. These are not only created for the extracted information, but also for possible candidates. A simple meta-feature, for example for the extraction of headlines, states that the bold feature indicates a headline in this document.

Confidence Estimation Since even only a single incorrect information can lead to a potentially high number of incorrect information, the correctness and confidence of an information is essential for the meta-level information extraction. There are two ways to identify information suitable for the extraction of meta-features. If the knowledge engineer already has certain assumptions about the content of the input documents, especially on the occurrence of certain information, then these expectations can be formalized in order to increase the confidence of the information.

Transfer Knowledge The transfer knowledge models the human behavior in practice and can be classified in three categories: *Agglomeration* knowledge processes multiple meta-features and creates new composed meta-features. Then, *projection* knowledge defines the transfer of the meta-features to possible candidates of new information. *Comparison* knowledge finally formalizes how the similarity of the meta-features of the original information and candidate information is calculated. The usage of these different knowledge types in an actual process depends on the kind of repetitive structures and meta-features.

This process model provides a generic framework for improving information extraction models in general. The actual implementation as a rule set can be engineered in an efficient manner and with minimal labor. The rules for the two domains are summarized in the following with pointers to the elements of the process model.

In the curricula vitae, the meta-features for the description of the consistency are based on the position of confident information dependent on the layout and in relation to other entities. Agglomeration knowledge uses these meta-features to formalize a pattern of the common appearance of the companies. Then, projection knowledge uses this pattern to identify new information that is rated by rules for confidence estimation. The actual rule set is quite straightforward. Some initial rules identify companies with high precision by using only confident

patterns relying on prefixes, suffixes or dictionaries for locations. Some of the companies, for example, are easily identified if they occur in a list of known companies, if there is a known keyword nearby (e.g., "Client:"), or if the location of the office is given afterwards. Then, the patterns of meta-features are initialized by joining the properties of these entities. The patterns may also include relation to other types of entities, e.g., a rule like "companies are located in the second line of the project after the mention of the date period". This knowledge is then projected to projects where no entities have been found yet.

In the discharge letters, the meta-features for the description of the consistency are based on the layout. The expectation of a "Diagnoses" or "History" (Anamnesis) headline is used to identify confident entities. Then, meta-features describing its actual layout are created and transferred by projection knowledge. Finally, comparison knowledge is used to calculate the similarity of the layout of the confident information and a candidate for a headline. The actual rule set is again quite straightforward. Some initial rules identify frequent headlines that are easily identified: headlines for "Diagnoses" or "History" (Anamnesis). Then, properties describing the layout of these headlines are stored in variables. These properties cover predefined features the font of the text fragment (bold, italic, underlined), the ending with a colon, or presence of free lines. After possible candidates for headlines are created, rules compare the properties of the confident headlines with the properties of each candidate. If they share the same properties, then a new headline was identified.

5.1.2 Experimental Results

For the evaluation of the precision-driven approach with improved recall by exploiting the context-specific consistencies, two rule-based prototypes were developed. The rules for information extraction in curricula vitae were engineered and tested on a very small dataset, which contains overall 15 documents with 72 companies. This minimal dataset is hardly suitable for an experimental evaluation, but it reflects the initial situation in a real-world project. An information extraction prototype should be developed as a feasibility analysis using this limited set of confidential documents. The rules were developed using five documents and the remaining ten documents were applied for testing. The development of the rules for identification of headlines in clinical discharge letters was based on a larger dataset, which contains overall 141 documents and 1515 headlines. About ten percent of the documents were used to engineer the rule set and the remaining documents were applied for testing. As a comparable model for both domains, a CRF was trained and tested on the respective collection of documents for testing the rule-based approaches. This setting leads of course to unrealistic predictions of the CRF model, but illustrates the advantages of the rule-based approach. The CRF was trained with the same features, the rules were based on.

The results of the experimental evaluation for both domains are given in Table 5.1. The rule-based approach achieved an F_1 score of 0.979 for the identification of companies in curricula vitae, whereas the CRF was only able to reach an F_1 score of 0.750. This inferior result of the CRF is mainly caused by the limited set of training examples and the restricted set of features. More important, even by testing on the training set, the CRF was not able to overfit on the examples. The rules, however, are able to extract almost all entities on unseen document with only minimal effort. The approach is very efficient and effective since the set of applied rules was written within one hour.

The rule-based approach achieved an F_1 score of 0.972 for the identification of headlines in clinical discharge letters, whereas the CRF was only able to reach an F_1 score of 0.871. The probabilistic model was again not able to reproduce the examples. The CRF extracted the same headlines as the rule-based approach in many documents. However, the conflicting layout styles of the same authors caused, as expected, a high number of false negative errors resulting in a lower recall value. The rule-based approach is not only very effective but also rather efficient, since it took only about 1-2 hours for engineering the necessary rules, considerably less time than the time spent for the annotation of the examples.

		Precision	**Recall**	**F_1**
Curricula Vitae	CRF	0.938	0.625	0.750
	Rules	1.0	0.958	0.979
Discharge Letters	CRF	0.979	0.785	0.871
	Rules	0.991	0.954	0.972

Table 5.1: Experimental results given for recall expansion during precision-driven prototyping in two domains. Rules implement the consistency-based approach for increasing the recall. Additionally, F_1 scores for a CRF model trained and tested on the test data using the same features are added [121].

5.2 Stacked transformations

The case study introduced in this section describes an iterative process model similar to the approach in the last section. The rules in this case study, however, are integrated in a more sophisticated process model where the created annotations are incrementally corrected based on a more complex model of consistencies. Furthermore, instead of processing semi-structured documents like curricula vitae or discharge letters, the segmentation of references in scientific publications is investigated. This approach shares some characteristics with error-driven transformations for part-of-speech tagging (cf. [34]), but applies the transformations based on conflicts concerning a model describing the occurring consistencies instead of patterns of tokens[47].

The aspects of context-specific consistencies in the domain of segmentation of references are shortly recapitulated for convenience (cf. Section 3.2.1). A document with references like scientific publications is often created in a single creation process, e.g., an author writes an article by using LaTeX. These documents contain similar styles, e.g., an author uses in a single document always the same layout for headlines or the used BibTeX style determines the appearance of the references in a paper. Patterns describing these similarities and regularities can be detected and used to improve the extracted information. However, patterns can vary strongly between different documents in a domain.

[47] The content of this section has been published in Peter Kluegl, Andreas Hotho, Frank Puppe. Local Adaptive Extraction of References. In 33rd Annual German Conference on Artificial Intelligence, Springer, 2010. [123]

5.2.1 Rule Sets

A common problem with extracting references is the different style resulting in conflicting patterns for global models, e.g., different separators between interesting fields or different sequences of the fields. This rule-based approach tries to apply a short term memory to exploit these context-specific consistencies. A simple information extraction component extracts instances and fields of information. The next phase, however, does not simply try to correct errors using a gold standard, but rather applies a local adaption step (adaptive IE component AIE). An analysis of the occurring patterns returns a set of conflicting information that can be corrected using transformation-based rules. This approach works best if the assumption holds that the document contains several instances of information and was created in the same process. The whole process is depicted in Figure 5.2.

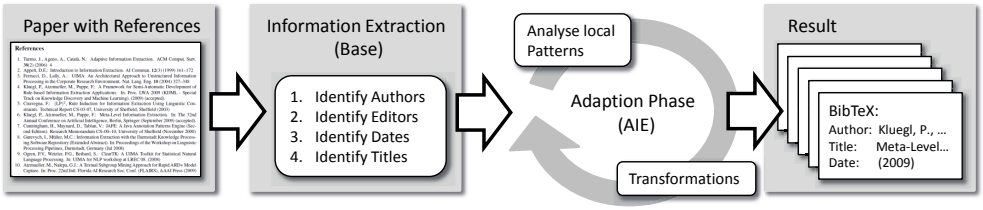


Figure 5.2: Overview of the adaptive process with initial information extraction, analysis step and transformation phase.

The base component consists of a simple set of rules for the identification of the *Author*, *Title*, *Editor* and *Date*. The given features are token classes and word lists for months, first names, stop words and keywords, e.g., indicators for an editor. After some reasonable rules were handcrafted, the adaptive component AIE is created and refined for the rules of the base component. The separators located at the beginning and the end of the BibTeX fields and the sequence of the fields are sufficient for a description of the applied creation process. Therefore, handcrafted rules aggregate the separators and sequences of the initially extracted information. They build a the local model by remembering the dominant separators and neighbors of each field. This local model is compared with all initially extracted information for an identification of possible conflicts. After this analysis phase, transformation-based rules are applied that utilize the local model and the information about conflicts in order to restore the consistency of the extracted information and therefore to correct the errors of the base component.

The approach is illustrated along the following example which is a typical text fragment representing a single reference of a reference section:

> Kluegl, P., Atzmueller, M., Puppe, F., TextMarker: A Tool for Rule-Based Information Extraction, GSCL Conference 2009, 2nd UIMA@GSCL Workshop, 2009

A reasonable result of the base component could consist in the following annotations:

> **<A>**Kluegl, P., Atzmueller, M., Puppe, F., TextMarker:**</A>** **<T>**A Tool for Rule-Based Information Extraction, GSCL Conference**</T>** **<D>**2009**</D>**, 2nd UIMA@GSCL Workshop, **<D>**2009**</D>**

The exemplary annotations contain four errors: "TextMarker:" is part of the *Author* and is missing in the *Title*. The *Title* and *Date* contain additional tokens of the conference. Inline annotations for the *Author* (<**A**>), *Title* (<**T**>) and *Date* (<**D**>) are used as a simple description:

The analysis of the document results in the following patterns describing the internal consistency of the references: Most of the identified *Author* and *Title* annotations end with a comma. The *Title* follows directly after the *Author* and the *Date* is located near the end of the reference. With this information at hand, conflicts for the first *Date*, the end of the *Author* and *Title* are identified. Several handcrafted transformation rules are applied to solve these conflicts. In the example, the *Author* and *Title* are both reduced to the last comma and only the last *Date* is retained. The following exemplary rule shifts the end of the *Author* by a maximum of four tokens to the next separator listed in the local model (*EndOfAuthor*) if a conflict was identified at the end of the *Author* annotation (*ConflictAtEnd*):

```
1 Author->{ANY{STARTSWITH(Author) -> SHIFT(Author,1,3)}
2   # EndOfAuthor ANY[0,4]? ConflictAtEnd;};
```

UIMA Ruta Listing 5.1: Exemplary rule for consistency-based correction of the author.

However, after these changes the *Title* does not directly follow the currently detected *Author* field. Therefore, the *Title* is expanded, which results in the following correct annotation of the example:

> <**A**>Kluegl, P., Atzmueller, M., Puppe, F.,</**A**> <**T**>TextMarker: A Tool for Rule-Based Information Extraction,</**T**> GSCL Conference 2009, 2nd UIMA@GSCL Workshop, <**D**>2009</**D**>

The transformation rules used to correct the errors of the base component are applied in a generic framework. They are completely independent of the specific local patterns detected for the current document. This is an immense advantage of the approach, because the same rules will work with different and unknown separators or field sequences. The complete rule set is linked in the respective publication [123].

5.2.2 Experimental Results

The experimental evaluation for this case study investigates the applicability of stacked transformations based on context-specific consistencies for the segmentation of references. At the time when the rule set was developed, there was no labeled dataset with multiple references originating from the same publication available.

For a comparable evaluation of the approach, the evaluation uses the commonly used Cora dataset (D_{Cora}: 500 references [153]). The dataset D_{Cora}^{All} contains 489 references missing line 100-109, which are utilized for the development of the Base component and one reference with damaged markup. However, this dataset is not directly applicable for the development and evaluation of the approach as the Cora dataset does not contain information about the original document of every reference which is needed to derive the context-specific consistencies. Therefore, a simple script was developed in order to reconstruct reference sections as they would

occur in real publications using only references of the available dataset. D_{Cora}^{Paper} contains 299 references of D_{Cora}^{All} in 21 documents and represents a selection of papers with a strict style guide applied. Due to the simplicity of the assignment script and the distribution of the reference styles in the dataset, a considerable amount of references could not be assigned to a paper. The dataset D_{Cora}^{Rest} contains the remaining 190 references. The dataset D_{Cora}^{All} is the union of D_{Cora}^{Paper} and D_{Cora}^{Rest}. The dataset D_{Dev} for the development of the adaptive component was created using 11 different publications and contains 213 references.

Table 5.2 contains the evaluation results of the simple rule-based component (Base) and the combination with the adaption phase (AIE). The token-level F_1 measure was applied on a single field and the instance accuracy was calculated for the complete reference. The AIE component was only applied on the dataset D_{Cora}^{Paper} which are 61% of all references. The Base component reached an average F_1 score of 0.953 and an instance accuracy of 85.4% on the development set D_{Dev}. The AIE component was tuned to achieve an average F_1 score of 1.0 and an instance accuracy of 100% on D_{Dev}.

	Base			AIE			Peng[160]	ParsCit[52]
	D_{Cora}^{Paper}	D_{Cora}^{Rest}	D_{Cora}^{All}	D_{Cora}^{Paper}	D_{Cora}^{Rest}	D_{Cora}^{All}	D_{Cora}	D_{Cora}
Author	0.984	0.983	0.984	0.999	0.983	0.993	0.994	0.990
Title	0.964	0.959	0.962	0.992	0.959	0.979	0.983	0.972
Editor	1.0	0.929	0.949	1.0	0.929	0.949	0.877	0.862
Date	0.981	0.958	0.973	0.999	0.958	0.983	0.989	0.992
Average	0.983	0.957	0.967	0.997	0.957	0.976	0.961	0.954
Instance	0.890	0.816	0.861	0.987	0.816	0.920	-	-

Table 5.2: Experimental results of the normal rule set (Base) alone and with the adaption phase (AIE) for the additional datasets. The results for the dataset D_{Cora}^{Rest} always refer to the Base component alone since no adaption was applicable.

The Base component yields results for all datasets which are comparable with knowledge-based approaches (cf. Section 3.2.1.3). The results of the machine learning methods are considerably better. Merely the score of the *Editor* field outperforms the related results and implies that rules seem to be very suitable for this task due to the long-range dependencies to the editor keyword. The overall lower performance for the *Date* field is caused by the fact that the development dataset contains no date information with a time span (e.g., dates like "20.-30. August 2011"). Hence, the Base component missed several true positives of the test datasets.

The adaptive approach is able to improve the accuracy of the initial rule set for both datasets, D_{Cora}^{Paper} and D_{Cora}^{All}. The results of D_{Cora}^{Rest} refer always to the result of the Base component since the context of the references is missing and the local adaption cannot be applied. The AIE approach achieves a remarkable result on the (D_{Cora}^{Paper}) dataset with an average F_1 score of 0.997. 98.7% of the references are extracted correctly without a single error. This is a reduction of the harmonic mean error rate by 88.2% for the complete reference sections. Errors in the extraction process can be observed for complicated references as well as for simpler ones. The presented approach is rather resilient to the difficulty of such references because the approach extracts

difficult references correctly by learning from other references of the same document. There was no adaption applied for the *Editor* field. The development dataset did not encourage any adaption of the *Editor*, because the Base component already achieved a F_1 score of 1.0 for this part on the development set D_{Dev}, resulting in a limited amount and quality of the necessary rules.

Although the adaption phase of the approach was only applied to 61% of the references of the dataset D^{All}_{Cora}, its evaluation results are able to compete with the results of Peng [160] and ParsCit [52]. The results are, however, generally difficult to compare to results from other researchers. First of all, the experimental evaluation used a transformed dataset, which required access to the labeled examples. The achieved outcomes are already very good and admit only marginally improvements for this dataset. The results of all three approaches were accomplished with different training or development datasets: A defined amount of references [160], a 10 fold cross evaluation [52] and an external dataset for the presented approach. The set of features applied, e.g., the content of the additional word lists, or even the tokenizer used to count the true positives may vary between different approaches. Besides that, it is difficult to compare a knowledge engineering approach with machine learning methods, because the knowledge engineer contributes an intangible amount of background knowledge to the rule set.

5.3 Usage in a Complete Application

The third case study for exploiting context-specific consistencies investigates the usefulness in a complete real-world application for processing clinical discharge letters. The application consists of a pipeline with six phases:

1. Conversion of the input document into a processable format. In the context of the application, the documents are given as word processing documents and are converted to HTML in order to retain the layout information.
2. Anonymization and deindentification using rules and dictionaries extracted from the clinical electronic health system.
3. Conversion of the HTML document into a plain text representation while retaining the layout information. This step facilitates the manual inspection of results in the available tooling.
4. Identification and classification of headlines and their corresponding segments into categories.
5. Information extraction in identified segments using specialized models for each kind of category (cf. [80, 79]).
6. Postprocessing of the extracted information and storage in a clinical data warehouse.

The focus of this case study lies on the fourth phase, which exploits the context-specific consistencies in order to improve the segmentation of the discharge letters. This part of the application was developed by Philip-Daniel Beck within the scope of his master thesis [13]. Due to the fact that the segmentation is a preprocessing step for further phases, high accuracy in

this step is an essential factor for the success of the complete application. The next sections will address the composition of the rule-based segmentation and the experimental results.

5.3.1 Rule Sets

The complete segmentation and classification concerning the fourth phase of the application is performed using UIMA Ruta rules. The general procedure that rules are implementing can be summarized by the following four phases:

1. Cutting out header and footer of the letter, and additional headers and footers of the pages.
2. Generating possible candidates for headlines.
3. Identification and classification of headlines using different approaches.
4. Extraction of categorized segments using the headlines.

The first phase is a preprocessing step, which ensures that only interesting parts of the document are processed. The header of the letter possibly contains text fragments that share a layout similar to that of headlines and thus aggravates the usage of rule-based approaches based on layout information. The footer of the letter should be removed or at least identified in order to secure the detection of the correct end of a segment in fourth phase. Headers and footers of a page do not contain interesting information and are removed in order to avoid problems and challenges in further processing. The last phase identifies the interesting sections by joining all text fragments from one headline to the next one or until the end of the document in case of the last section. The category of the section is determined by the category of the corresponding headline. The subheadings with similar layouts are a general problem and are dealt with by the combination of the different approaches. The second and third phases are described in more detail in the following sections.

5.3.1.1 Generating Candidates

The identification of possible candidates is an important part of this rule-based approach, because the remaining steps mainly only filter the set of candidates for identifying the headlines. The goal of the candidate generation is therefore obtaining a high recall so that no potential headlines will be missed. The rules for creating the candidates make use of the layout information given by the HTML document, punctuation marks that indicate the end of a sentence, and additional rules for dates and ICD10 codes. Periods and colons that are part of dates or abbreviations, for example, will be treated differently by the rules. A candidate typically begins with the start of a paragraph, but only if also a suitable end was identified. This end of a candidate is determined by the first applicable rule of the following list, which is extended with additional pointers to the corresponding examples in Figure 5.3:

- If the candidate uses the same layout for the complete span, ends with a colon, and possesses at least a highlighting like a bold, underlined or italic font, the end of the candidate is set to the end of the paragraph (cf. Type 1 in Figure 5.3).

- If the candidate is consistently formatted, possesses at least one kind of highlighting, but contains no colon indicating the end of a sentence, then the end of the candidate is again set to the end of the paragraph. This rule is a specialization of the first one for covering candidates before the remaining rules are applied.
- If the beginning of a candidate is formatted and a change of layout occurs near a colon, then the candidate ends with this colon (cf. Type 2 in Figure 5.3). Text fragments in parentheses without highlighting are ignored (cf. Type 3 in Figure 5.3).
- The candidate ends with a change of layout even if there is no colon.
- The candidate ends with the first colon (cf. Type 4 in Figure 5.3).
- If there is a highlighted text fragment near the begin of the paragraph, then the candidate ends with the next change of layout (cf. Type 5 in Figure 5.3).
- The end of the candidate is set to the end of an overlapping HTML headline, e.g., "</h1>".

Type 1	**Diagnosen:**
Type 2	**Aktueller Status:** Am Vorstellungstag Kopfschmerzen, ansonsten bis auf Beschwerden im li. Knie weitgehend o.B..
Type 3	**Laborbefunde vom DDD** (siehe Anlage): Krea 1,5 mg/dl, BUN 35 mg/dl, HS 10,0 mg/dl, Ca 3,7 mmol/l, K, 4,2 mmol/l.
Type 4	**Labor: Gesamtcholesterin 243 mg/dl, LDL-Cholesterin 149 mg/dl**
Type 5	Nach der **körperlichen Untersuchung** hat sich herausgestellt...

Figure 5.3: Five example of different types of headlines in German discharge letters. For each example Type 1-5, the complete paragraph is given [13].

This process still generates a huge amount of possible candidates, which should optimally cover all potential headlines. The set of candidates can, however, be safely reduced by removing candidates that fulfill certain conditions. The candidates are pruned with the rules in Listing 5.2. The rules investigate each candidate ("HC") and remove it if it contains more than 35 words, contains more than 16 words and does not end with a colon or are not highlighted, contains more than 8 words, does not end with a paragraph and contains a period indicating the end of a sentence.

```
HC{HC.numW>8,-FEATURE("endswithPar"),CONTAINS(PeriodNotOK)
  ->UNMARK(HC)};
HC.numW>16{-IS(Formatted)->UNMARK(HC)};
HC.numW>16{-FEATURE("endswithColon",true)->UNMARK(HC)};
HC.numW>35{->UNMARK(HC)};
```

UIMA Ruta Listing 5.2: Rules for pruning the set of potential candidates.

5.3.1.2 Properties of Headlines

The different rule-based approaches for identifying headlines in the set of candidates use mostly a limited set of properties or features. Table 5.3 gives an overview of the boolean attributes of a candidate, which reflect its structure and layout.

Property	Abbreviation	Description
endswithPar	*eP*	true if candidate ends with a paragraph annotation.
startswithPar	*sP*	true if candidate starts with a paragraph annotation.
short	*Sh*	true if the candidate contains only up to two words.
isBold	*Bo*	true if the candidate uses a bold font.
isUnder	*U*	true if the candidate is underlined.
isItalic	*I*	true if the candidate uses an italic font.
hasBackground	*Ba*	true if the candidate uses a specific background color.
endswithColon	*eC*	true if the candidate ends with a colon.
isHTMLHeadline	*H*	true if the candidate is a HTML headline (h1-h6).
containsDate	*D*	true if the candidate contains a date.
partOfCollection	*Co*	true if the candidate is part of an enumeration or table.

Table 5.3: Listing of boolean properties of the candidates [13].

This set of properties can directly be applied for identifying the headlines by interpreting them individually as rules. This approach may not be sufficient for the application, but grants insights in the composition of the dataset. Table 5.4 provides an overview of the coverage of the properties in the development set used during the engineering of the rules (cf. Section 5.3.2). The results indicate that most headlines end with a colon and are formatted using a bold font. Headlines with an italic font or with a different background are rather sparse. The most accurate classification is achieved by the combination of both properties (colon and bold) with an F_1 score of 0.860. While this seems to be an already good result, it is by far not sufficient in the scope of the application.

5.3.1.3 Score-based Approach

Score-based approaches assign a numerical score for each occurring property to a candidate. If the aggregated sum of the scores for one candidate exceeds a given threshold, then the candidate is identified as a headline. Given the properties specified in the last section, a vast amount of

Property	Recall	Precision	F_1	TP	FP	FN
hasBackground	0.032	0.788	0.062	115	31	3432
isBold	0.932	0.707	0.804	3307	1369	240
containsDate	0.178	0.472	0.258	630	705	2917
endswithColon	0.981	0.576	0.726	3479	2559	68
endswithPar	0.759	0.629	0.688	2692	1590	855
isItalic	0.005	0.016	0.080	19	1208	3528
isUnderlined	0.507	0.504	0.506	1799	1771	1748
isBold $\wedge$ *endswithColon*	0.918	0.808	0.860	3257	772	290
isBold $\wedge$ *endswithPar*	0.724	0.766	0.744	2567	783	980
short	0.707	0.525	0.603	2509	2270	1038

Table 5.4: Classification of candidates as headlines using individual properties or combinations of two properties. The results are given for the development set [13].

possible configurations exist. Table 5.5 contains ten different variants that apply different scores for each property and also define a suitable threshold. A candidate like Type 1 in Figure 5.3, for example, will achieve a score of 2+2+4+2+0+0+0+2+0+0+0=12 for variant Scoring_{10} and will be classified as a headline.

Variant	eP	sP	Sh	Bo	U	I	Ba	eC	H	D	Co	Threshold
Scoring_1	1	0	1	1	1	0	0	1	1	0	0	2
Scoring_2	1	0	1	1	1	0	0	1	1	0	0	3
Scoring_3	1	0	1	1	1	0	0	1	1	0	0	4
Scoring_4	2	2	2	3	1	1	1	4	1	-1	-1	8
Scoring_5	2	2	2	2	1	1	1	5	1	-1	-1	10
Scoring_6	2	2	2	2	1	1	-1	3	1	-1	-1	8
Scoring_7	2	2	2	2	1	1	1	2	1	-1	-1	8
Scoring_8	3	2	3	4	1	1	1	2	1	-1	-1	10
Scoring_9	2	2	2	4	1	1	1	2	1	-1	-1	8
Scoring_{10}	2	2	4	2	1	1	1	2	1	-1	-1	8

Table 5.5: Overview of variants with varying scores for the different properties [13]. The results of each variant are given in Table 5.7

5.3.1.4 Keyword-based Approach

The keyword-based approach applies an extensive list of regular expressions covering indicator words and their variants. This approach is not only applied for identifying headlines in the set of candidates, but also for assigning a predefined category to the identified headline. The rules define furthermore strong categories, a concept that is utilized by other approaches, and aggregates a few kinds of categories. A more detailed description of this approach and the

categories can be found in [13]. Overall, this approach is similar to the segmentation component provided in cTAKES (cf. Section 3.2.3.3).

5.3.1.5 Consistency-based Approach

This rule-based approach exploits the context-specific consistencies in the document by comparing the layout of the candidates to the layout of confident headlines. The headlines should be easily identified in the set of candidates and should occur preferably in each document. To these headlines belong diagnoses, history (anamnesis) and lab data. If a candidate possesses the identical values in the properties *startswithPar*, *isBold*, *isUnderlined*, *isItalic*, *hasBackground* and *endswithColon* as a confident headline, then it is classified as a headline. This approach is strongly inspired by the prototype described in Section 5.1.1.

5.3.1.6 Correction-based Approach

The last approach applies transformation-based rules for covering a small set of misclassified candidates. Some candidates can only be correctly identified by the previous approaches with considerable efforts. Transformation-based rules can here be easily applied for correction. Examples for these correction rules and also rules implemented for the other approaches can be found in the master thesis [13].

5.3.2 Experimental Results

The application for segmenting and categorizing sections in clinical discharge letters was created using a labeled dataset. The dataset contains overall 700 gold standard documents manually labeled by Philip-Daniel Beck of which 500 documents have been used for developing the rule sets and the remaining 200 documents for the evaluation of the different approaches. The experimental study was designed to compare the performance of the different approaches described in the previous section individually and also combinations of them. The results in this section provide always the performance for the test set and also for the development set. This grants insights on how far the rule sets have been optimized on the development set and if there are major discrepancies between the results on the development set and the test set.

Before the actual approaches for the identification of the headlines are evaluated, a closer look is taken on the generation of possible candidates. Table 5.6 contains the results for the development set and the test set. The approach was able to fulfill the requirements by achieving a high recall. Two headlines have been missed in the test set, which will also not be identified as headlines by the following approaches. One false negative was caused by an error in the rule set, and the second missed headline did not start with a paragraph.

Candidates	**Recall**	**Precision**	$\mathbf{F_1}$	**TP**	**FP**	**FN**
development set	1.0	0.482	0.651	3547	3812	0
test set	0.998	0.594	0.745	1239	845	2

Table 5.6: Results of candidate generation [13].

Table 5.7 depicts the experimental results for the different variants of the scoring approach. The rules and the rating were developed to achieve a high recall, which explains the discrepancy between the development set and the test set. The results are overall mediocre. The best variant on the development set ($Scoring_7$) was able to achieve an F_1 score of 0.817 while producing one of the worst predictions on the test set. The best F_1 score of 0.880 on the test set was achieved by variant $Scoring_5$, which defined a high score for candidates that end with a colon. The scoring-based approach performs, therefore, too poorly in the scope of the application.

Scoring		**Recall**	**Precision**	**F_1**	**TP**	**FP**	**FN**
$scoring_1$	development set	0.996	0.541	0.701	3532	2993	15
	test set	0.986	0.680	0.805	1224	575	17
$scoring_2$	development set	0.932	0.623	0.747	3306	1998	241
	test set	0.928	0.783	0.849	1152	320	89
$scoring_3$	development set	0.997	0.539	0.700	3536	3023	11
	test set	0.986	0.675	0.802	1224	588	17
$scoring_4$	development set	1.0	0.528	0.691	3546	3174	1
	test set	0.948	0.768	0.849	1177	355	64
$scoring_5$	development set	0.987	0.560	0.714	3500	2752	47
	test set	0.909	0.853	**0.880**	1228	194	113
$scoring_6$	development set	0.963	0.570	0.716	3414	2572	133
	test set	0.546	0.880	0.674	677	92	564
$scoring_7$	development set	0.906	0.743	**0.817**	3213	1109	334
	test set	0.745	0.871	0.803	925	137	316
$scoring_8$	development set	0.983	0.574	0.725	3495	2599	52
	test set	0.893	0.804	0.846	1108	270	133
$scoring_9$	development set	0.998	0.561	0.718	3538	2769	9
	test set	0.922	0.788	0.850	1144	308	97
$scoring_{10}$	development set	0.990	0.560	0.716	3513	2756	34
	test set	0.881	0.770	0.822	1093	327	148

Table 5.7: Results of the different variants based on scoring rules [13].

Table 5.8 provides an overview of the results of the keyword-based approach. The rules are able to achieve a remarkable recall thanks to the general character of the approach. The applied rules identify all variants of word combinations that are usually used in headlines, which explains also the lower precision. The recall for the test set decreases due to the fact that the regular expressions have been designed based on the development set. Overall, the approach is able to achieve an F_1 score of 0.912 on the test set, a considerable higher score than the score-based approaches.

Keyword	**Recall**	**Precision**	F_1	**TP**	**FP**	**FN**
development set	0.997	0.845	0.915	3536	647	11
test set	0.988	0.847	0.912	1226	221	15

Table 5.8: Results of keyword-based headline identification [13].

Table 5.9 contains the results of the consistency-based approach. The rules were able to achieve a remarkable precision both on the development set and test set. This result is a clear indicator that the authors of the discharge letters apply the layout of headlines not for other text fragments. However, the recall provides opportunities for improvement. By achieving an F_1 score of 0.960 on the development set and an F_1 score of 0.949 on the test set, this approach outperforms the other approaches considerably.

Consistency	**Recall**	**Precision**	F_1	**TP**	**FP**	**FN**
development set	0.923	0.999	0.960	3275	4	272
test set	0.906	0.996	0.949	1124	5	117

Table 5.9: Results of consistency-based headline identification [13].

After evaluating the approaches individually, the next results provide insights on the performance of the combinations of the approaches.

The results given in Table 5.10 display the performance of the combination of the score-based and keyword-based rules. The usage of scored candidates reduces the recall only marginally, but improves the precision considerably. The rules are able to achieve an F_1 score of 0.988.

Score+Keywords	**Recall**	**Precision**	F_1	**TP**	**FP**	**FN**
development set	0.994	0.982	0.988	3524	63	23
test set	0.985	0.992	0.988	1222	10	19

Table 5.10: Results of score-based headline identification combined with keywords [13].

By additionally using the rules based on consistencies, the F_1 score is increased by 0.003, which constitutes an error reduction of about 25% (cf. Table 5.11).

Score+Keywords+Consistency	**Recall**	**Precision**	F_1	**TP**	**FP**	**FN**
development set	0.994	0.982	0.988	3526	66	21
test set	0.991	0.990	0.991	1230	12	11

Table 5.11: Results of the combination of the score-based, keyword-based and consistency-based headline identification [13].

The further usage of correction-based rules leads to an F_1 score 0.992 on the test set (cf. Table 5.11). While this is only a small improvement, the minor labor for writing these rules can be ignored.

Score+Keywords+Consistency+Correction	**Recall**	**Precision**	**F_1**	**TP**	**FP**	**FN**
development set	0.996	0.992	0.994	3531	29	16
test set	0.991	0.994	0.992	1230	8	11

Table 5.12: Results of the combination of the score-based, keyword-based and consistency-based headline identification with additional correction rules [13].

The combination of the approaches provides a high-quality method for the segmentation and categorization of clinical discharge letters. Overall, 93.5 percent of the documents haven been correctly processed without one single error. A detailed error analysis can be found in the master thesis [13].

5.4 Discussion

One advantage of rule-based information extraction approaches is the fact that the knowledge engineer is able to keep improving the rule set in order to increase its accuracy. This process is, however, time consuming and more and more labor and effort needs to be invested for further improving the performance of the rule set. A central goal for rule-based system is therefore the ease of the engineering work. This can be achieved by good tooling support, a clear and powerful rule language, and a strong documentation. The first case study takes another path by introducing a novel methodology for engineering that allows the user to create effective rule sets in a extremely efficient manner. The developed rule sets can be summarized as precision-driven prototypes with additional recall expansion by exploiting context-specific consistencies. This idea enables to achieve results almost ready for production if the assumption of the consistent composition of entities holds. The experimental evaluation in this case study showed the advantages of the approach: The knowledge engineer is able to create rapid prototypes with minimal labor ranging under two hours of work. These kinds of prototypes are perfect for feasibility analysis in domains with context-specific consistencies. Furthermore, these prototypes achieve results good enough for speeding up the development of the actual applications as it was done in the third case study for the segmentation of discharge letters.

The second case study of the segmentation of references applied a more sophisticated approach. While the prototypes in the first case study can be interpreted as stacked phases, the rules in this case study clearly separate the phases by introducing a distinct model of consistency. The properties of the identified entities are analyzed and aggregated to a non-trivial description of the consistencies. The framework of this process is more domain independent than the rather simple approach of the first case study. The idea of just correcting the inconsistent entities is overall also more flexible and can be combined with arbitrary components in a more efficient manner. The combination is of course also possible with the approach for increasing the recall, but here only new entities can be identified. It is conceptually more efficient to apply transformations, which are able to remove the errors produced by the first component. The

experimental evaluation of this case study illustrated the advantages of the approach by achieving remarkable results. These results represent, however, only a feasibility analysis due to the limited expressivity of the applied dataset. In real-world scenarios, slatternliness and other reasons introduce inconsistent compositions of entities, which did not occur in the partition with the stand out results. Nevertheless, it is a remarkable result that a knowledge based approach can even compete with probabilistic models in this domain (cf. Section 3.2.1.3).

The last case study shows the usefulness of exploiting context-specific consistencies in a complete real-world application. It tries to combine various rule-based approaches and does not only rely on one approach like the other case studies. The experimental study shows that the consistencies are also able to improve a thorough engineered application. When the consistency-based rules are applied separately, they are able to achieve a remarkable precision. In combination with other rule-based approaches, it is still able to improve the accuracy and even increases the recall. The absolute improvement seems not substantial in the first place, but it is very important in critical applications like processing medical documents, and especially if the results are used further in a pipeline. In the experience of the author, it is hard to gain any more percent points in this region of F_1 scores. The consistency-based approach accomplishes this improvement while not requiring the equivalent effort of a common rule-based approach. The relative error reduction concerning the F_1 score of 25% can be considered a realistic improvement for exploiting context-specific consistencies in real-world scenarios.

Overall, this section presented three approaches for exploiting consistencies in three different domains. The first and last case study utilizes a simple description of the consistency to find additional entities and to filter inconsistent entities, respectively. The second case study applies transformations for correcting the given entities. The rules change entities in order that they become consistent with the dominant composition. All three approaches have proven their usefulness and provide an efficient alternative to common engineering patterns.

Chapter 6

Machine Learning Approaches

Machine learning approaches to information extraction are often assumed to be superior to rule-based approaches, especially for well-defined and stable specifications and when enough labeled data is available. The last Chapter 5 described a few case studies of rule-based applications that exploit context-specific consistencies. These rule sets are inherently domain-specific and only the general strategy but not the actual functionality can be utilized for solving information extraction tasks in other domains. This section describes machine learning approaches that provide a more generic framework for exploiting the context-specific consistencies. They are applicable for all domains where the assumption about the similar composition of entities holds. The chapter starts with an introduction how a model for the consistencies in a document can be learnt. This model is utilized by different variants of Conditional Random Fields in order to achieve superior results in the targeted domains. The information extraction task concerning the segmentation of discharge letters was already covered by rule-based approaches resulting in a extremely high accuracy so that hardly any improvement can be achieved using probabilistic models. Thus, the experimental studies for evaluating the machine learning approaches consider only segmentation of references and information extraction in curricula vitae[48].

6.1 Learning Context-specific Consistencies

The rule-based approaches for exploiting context-specific consistencies in Chapter 5 follow a straightforward strategy. They explicitly specify the consistent properties of the entities with rule-based methods by simply selecting the dominant composition or even only single characteristics. The search space for finding the consistencies is therefore directly engineered and does not support much adaption beyond the boundaries of the considered properties during rule execution. The context-specific consistencies can, however, also be automatically induced using machine learning methods. First, the process of applying binary classifiers for learning

[48] Parts of the contents of this chapter have been published in three research papers: Peter Kluegl, Martin Toepfer, Florian Lemmerich, Andreas Hotho, and Frank Puppe. Stacked Conditional Random Fields Exploiting Structural Consistencies. In Pedro Latorre Carmona, J. Salvador Sanchez, and Ana Fred, editors, Proceedings of 1st International Conference on Pattern Recognition Applications and Methods (ICPRAM), pages 240–248, Vilamoura, Algarve, Portugal, February 2012. SciTePress [128]. Peter Kluegl, Martin Toepfer, Florian Lemmerich, Andreas Hotho, and Frank Puppe. Exploiting Structural Consistencies with Stacked Conditional Random Fields. Mathematical Methodologies in Pattern Recognition and Machine Learning. Springer New York, 2013. 111-125 [129]. Peter Kluegl, Martin Toepfer, Florian Lemmerich, Andreas Hotho, and Frank Puppe. Collective information extraction with context-specific consistencies. In Peter A. Flach, Tijl De Bie, and Nello Cristianini, editors, ECML/PKDD (1), volume 7523 of Lecture Notes in Computer Science, pages 728–743. Springer, 2012 [127].

description of consistent compositions of the entities is introduced. This process is then illustrated by an example for segmentation of references. The usefulness of the presented approach is evaluated in two scenarios. The first setting introduces random errors for highlighting the ability of the approach to identify the correct boundaries using erroneous data. The second setting investigates the identification of boundaries based on realistic predictions.

6.1.1 Modeling Consistencies with Classifiers

For exploiting context-specific consistencies it is very helpful to acquire a description or model for the consistencies in each context or document. A model describing the consistent entities is not only able to point out inconsistent compositions, but it allows one also to identify additional entities that share a consistent composition. The identification of consistent and inconsistent entities can be divided into six steps:

1. Determine type of description for the consistencies.
2. Select classifier for learning the consistencies.
3. Provide a prediction of the entities.
4. Create a dataset for the classifier.
5. Learn classifiers on the dataset.
6. Apply classifiers on the dataset.

The first two steps are general decisions about the actual approach that should be used for modeling the consistencies. They cover the representation of possible consistencies and the algorithm applied for learning the model. These decisions have to be made during the creation of the information extraction application. They depend on the domain and the appearance of the consistencies. The remaining four steps are performed when processing a document. They include preprocessing steps, the induction of a description for the consistencies and the usage of the description for identifying consistent and inconsistent entities. The next step would exploit this information in an information extraction model in order to improve the accuracy in domains where the assumptions about the consistent composition holds. Approaches for this step are described in Section 6.2 and Section 6.3. Both approaches rely on sequence labeling methods and in particular on Conditional Random Fields. These graphical models unroll a graph of random variables for the tokens. The assignment of the random variable specifies identified entities. The six steps for modeling the context specific consistencies are described in the following sections. Since the gained information should be exploited by Conditional Random Fields, the description of the steps point out the connection to the respective concepts of graphical models, e.g., token sequence and label sequence.

6.1.1.1 Determine Type of Description for Consistencies

The different possibilities to describe the composition of entities have already been investigated in Section 3.1. The developer needs to select one type or a set of types according to the domain, in which information extraction is performed. The different types are repeated for convenience:

- Transition between entities
- Boundaries of entities
- Content of entities
- Order of entities

The transition between entities is the most generic approach for describing the composition of entities. It can also be utilized to model the other types. The concept is similar to the representation of entities in sequence labeling methods, which rate different features for a specific combination of two labels. When the entities are projected on the token sequence of the document, the transition from one token to another can be used to describe specific aspects of the involved entities. This description can be extended with additional properties and features that occur near the two tokens. Tokens of different entities lead to a description of the boundaries of entities and tokens of the same entity represent a statement about the content of the entities. This type of description is well suited for domains where the properties of the composition depend on the neighboring entities. In the domain of segmentation of references, the composition of the entities sometimes depends on combination of the entities. While the pages, for example, begin with a keyword like "pp." in references for publications in proceedings, the entity directly starts with the first page number for articles published in journals. Even if the exact kind of publication is probably unknown when processing a reference section, the combination of entities or labels for tokens can be utilized to achieve the same differentiation. The pages are located after a booktitle entity in references concerning proceedings, but they are following a volume entity in the references to journals. When the transition between entities is applied for describing the composition, it is still possible to provide consistent descriptions since the pages entities are not modelled directly. Instead the transition between booktitle and pages is modelled with the property of a special keyword and the transition between volume and pages is described with a different property, e.g., the occurrence of a number.

The boundaries of the entities can be used as a more robust description in many domains. They model the properties of the first and last token of the entities. Thus, boundaries are a special case of transition between two different kinds of entities. Descriptions using the transition between entities provide some advantages, but there are also disadvantages especially in domains like curricula vitae. Here, the entities are sometimes located next to each other, but sometimes there are also additional tokens between them. Transition-based descriptions would need to model transitions between an entity and the following entity, but also from an entity to tokens that are not part of an entity. This aggravates the learning process described later due to the reduced amount of examples. It is hardly possible, for example, to induce a valuable and useful description of consistent compositions from one example. Descriptions using only the boundaries of entities do not suffer from this disadvantage. Here, the first and last tokens are described independently of the previous or following entity.

Descriptions for the content of the entities are similar to descriptions based on boundaries. This type is again a special case of the transition-based description since it models the transitions between tokens of the same kind of entity. Descriptions for the content are useful if the consistencies occur for all tokens of an entity. Examples for this consistency are bold or underlined headlines. These properties occur typically also at the beginning or the end of an entity. However,

content-based descriptions are potentially more robust especially if all tokens provide consistent features but not the tokens concerning the boundaries. Furthermore, providing information about each token of an entity instead of only the first and last token is able to provide more evidence for graphical models that rely, for example, on a restricted Markov order.

The fourth type of description for the consistent aspects of the entities' composition is the order of the entities. In contrast to the other types, this description does not use properties or features, but only the sequence of entities. In some domains, the entities in a document may not share any similarities that can be specified with the available set of features. However, the order the entities occur may still be consistent within one document. This information can be exploited similar to the other kind of description in order to gain knowledge about the correct appearance of entities.

6.1.1.2 Select Classifier for Learning Consistencies

Context-specific consistencies can be induced by a large amount of different approaches. A simple approach includes similarity measures that rate the consistency of different aspects of the entities. An example for using mutual information can be found in Toepfer et al [194]. This work applies binary classifiers trained in a supervised fashion in order to learn models for identifying consistent and inconsistent parts of the entities. This section discusses properties of a classifier suitable for this task. How the classifier is applied for learning the context specific consistencies is described in the following section.

Any kind of binary classifier trained in a supervised scenario can be applied for gaining information about the consistencies in a document. There are however specific properties a classifier should fulfill in order to achieve best possible results in this task. These properties include the following features:

Generalization The classifier should not tend to overfit since it is trained and applied on possibly erroneous data. These errors should not be reproduced. The learning algorithm needs to support a certain level of generalization, which is able overcome the dirty input data. The generalization ability of the classifier is the main functionality that enables to detect consistent aspects of the entities. In general, generalization can be improved and overfitting can be restrained by limiting the amount of attributes.

Hypotheses space The hypotheses space of the classifier need to be compatible to the consistencies in the domain. The classifier should not use different hypotheses in order to solve the classification problem if only one consistency exists. Classifiers combining multiple hypotheses tend to model consistent examples as well as the inconsistent examples in the given training data. This results in a deteriorated generalization concerning the given task.

Label bias The classifier should handle label bias correctly. The given training data consist of a few true positive examples and a large amount of true negative examples. If the learning algorithm only tries to optimize the accuracy, then it is able to achieve already good results when it classifies all examples as true negative. This classification prevents any attempts to identify the consistencies.

Performance The classifier should be efficient with respect to its runtime performance. A classification model for each kind of description needs to be learnt and applied during processing the current document. While the application of the classifier on the data is typically fast, the learning of the model should be fast enough for the targeted applications. The runtime performance can be improved for many classifier implementation, e.g., by reducing the amount of attributes. The set of examples, however, should not be limited in order to speed up the learning process.

Many classifiers fulfill these requirements more or less. Support vector machines (SVM) [49] are one of the most popular binary classifiers and can be applied for this task. They produce, however, models that are hardly interpretable by humans, which aggravates the development and tuning of the classification task. SVM may be more powerful than other classifiers and are able to achieve better result in many classification tasks. This task differs, however, from typical classification scenarios since the learnt model is applied on the training data in order to identify consistent and inconsistent examples. Classification models which are interpretable by humans often rely on rules. Ripper [47] is probably the currently most popular learning algorithm for inducing a set of rules. Ripper learns rules one at a time by growing and pruning each rule and then adds them to a result set until a stop criterion is met. After adding a rule to the result, examples covered by this rule are then removed from the training data. Ripper is known to be on par regarding classification performance with other learning algorithms for rule sets, e.g., C4.5 [165], but is computationally more efficient. However, an application of Ripper for the given task suffers from one of the main characteristics of the learning algorithm. By trying to cover all examples, the algorithm tends to learn rules for inconsistent examples. Therefore, Ripper is not suitable for domains in which the entities share only one kind of composition. It is rather useful in scenarios where different manifestations of consistencies may occur for one kind of entity. An example for this is the domain of reference segmentation combined with a boundary description of the pages entity (cf. Section 6.1.1.1). This work utilizes subgroup discovery [117] for learning a model of the consistent composition of the entities. This approach fulfills all requirements since the generalization functionality can directly be controlled by the given amounts of attributes. The hypothesis space confirms with the assumptions in the targeted domains, e.g., only one subgroup is learnt to describe all entities of one kind, and no label bias occurs. Subgroup discovery is rather a supervised pattern discovery algorithm than a rule learning approach for classification. Finally, the runtime performance can be improved either by reducing the amount of attributes or by using optimistic estimates. The experimental evaluation in [128] investigated all three classifiers for learning the consistencies with the result that subgroup discovery outperformed the other classifiers. Thus, this work only investigates subgroup discovery for learning context-specific consistencies. The remainder of this section introduces subgroup discovery and optimizations for the given task.

Subgroup Discovery [117] (also called Supervised Descriptive Rule Discovery or Pattern Mining) performs an exhaustive search for the best k conjunctive patterns in the dataset with respect to a pre-specified target concept and a quality function. In the context of this work, only the best pattern is of interest. The most popular quality functions for a pattern p can be formulated in the following form:

$$q_a(p) = n^a \cdot (t - t_0), a \in [0;1] \tag{6.1}$$

n is the number of instances covered by this pattern, t is the share of the target concept for the respective pattern, t_0 is the target share in the total population and a is a parameter that can be used to trade a larger coverage of the rule with a higher deviation in the target share.

Additionally, different constraints on the resulting patterns can be applied, e.g., on the maximum number of describing attribute value pairs or the minimum support for a rule. While the resulting rules are not intended to be used directly as a classifier, a related approach using patterns based on improvement of the target share and additional constraints has been successfully applied as an intermediate feature construction step for classification tasks [11].

There are many possibilities to adapt subgroup discovery to the intended task of this work. The restriction of the maximum number of attributes of one subgroup or rule directly controls generalization. The rules are normally not able to overfit on the given examples with limited amount of attributes or features. The algorithm provides straightforwardly a score for each subgroup. This enables the usage of a threshold for pruning poor-quality descriptions, which potentially deteriorate the actual information extraction models. The most important adaption to the given task is the selection of the quality function, which allows further optimizations beyond the functionality of common classifiers. The Equations 6.2, 6.3 and 6.4 contain the utilized quality functions. Equation 6.2 represents the common F_1 measure. The subgroup discovery task tries to identify descriptions that reproduce the given examples as good as possible. The values tp, fp and fn refer to the true positives, false positives and false negatives when the examples covered by the subgroup are compared to the actual data.

$$F_1 = \frac{2 \cdot tp}{2 \cdot tp + fn + fp} \tag{6.2}$$

$$F_1^{exp} = F_1 \cdot \left(1 - \left(\frac{|tp + fp - E_y|}{\max(tp + fp, E_y)}\right)\right) \tag{6.3}$$

$$F_1^{exp2} = F_1 \cdot \left(1 - \left(\frac{|tp + fp - E_y|}{\max(tp + fp, E_y)}\right)^2\right) \tag{6.4}$$

The Equations 6.3 and 6.4 contain two optimized quality functions for learning the consistencies if additional information about the entities are given. The left part of these measures describes the traditional F_1 measure that is how well the pattern reproduces the given examples. The right factor is a penalty term for the divergence of the amount of covered examples to a given variable E_y, which is the expected amount of occurrences in a context. These quality function can greatly improve the given task since subgroups are preferred that identify the expected amount of boundaries or transitions. In the domain of reference segmentation, for example, one expects that each reference contains exactly one author. Although this is not true in general, it provides for a valuable weighting of the hypothesis space and further reduces overfitting.

6.1.1.3 Provide Prediction of Entities

Learning context-specific consistencies relies on a prediction of the entities or, in terms of graphical models, on a prediction of the label sequence. The different types of descriptions for consistencies have continuously mentioned the entities for referring to their boundaries or other aspects. The entities are, however, the output of information extraction and are of

course not available when processing the document. Therefore, some sort of prediction of the entities has to be provided for learning the context-specific consistencies. The prediction is inherently defective since otherwise no information extraction needs to be applied after all. The provided entities do not need to be perfect. They are only used to analyze the consistencies of their composition so that another model is able to extract the entities more accurately.

There are plenty of options for providing an initial prediction of the interesting entities. Basically, all approaches can be applied for this task. The most useful prediction depends on the domain and the available resources. In the domain of curricula vitae, for example, dictionary matching is already able to provide some initial entities. The approaches and techniques in this chapter should, however, be independent of the domain. Thus, the prediction is given by a simple baseline Conditional Random Field that is trained on a representative collection of examples. The model is utilized during processing of the document for extracting the necessary entities. Since the later approaches in Section 6.2 and Section 6.3 use also Conditional Random Fields for the actual information extraction, the overall approach is similar to stacked graphical models (cf. [131, 134, 128]).

6.1.1.4 Create Dataset for Classifier

After a prediction of entities is given, the dataset for training the classifier can be created. The composition and layout of the dataset depends on the format required by the classifier. This work utilizes subgroup discovery for learning the context-specific consistencies. Thus, the creation of the dataset is described using the terminology and concepts of this algorithm.

A new dataset needs to be provided for each document or context that is processed. Each token of the document will become a training example of the dataset. The attributes of each example consist of two separate sets. A set of binary attributes cover the available features, e.g., the features also utilized by the Conditional Random Fields. For performance reasons and to reduce the search space of the subgroup discovery task, the set of features is pruned and only relevant properties are retained. The set of attributes can also be extended by including features of nearby tokens in a fixed window, which avoid the limitations of classification techniques for sequences. The attribute "WORD@+1=pp" indicates, for example, that the next token equals the string "pp". The values of the attributes are set to true, if the feature occurs at the current token and false otherwise.

The second set of binary attributes represents the target attributes. They are calculated using the predicted set of entities. One attribute is added for each kind of entity and each kind of description used the model the consistencies. The set of entities does not have to cover all entities that should be extracted, but only the entities that are assumed to fulfill the expectation of consistent composition. When working with the boundaries of entities, this leads to two attributes for each type of entity, that are an attribute for the beginning of that kind of entity and one attribute for the end. The values of these attributes are set to true, if the token corresponds to the type of description, and to false otherwise. In the example of boundaries, the attribute for the beginning of an entity is set to true for each first token of this type of entity. When working with transition-based descriptions, the value is set to true in case a transition of the respective labels or entities occurred.

6.1.1.5 Learn Classifiers on Dataset

The next step after creating the dataset for the current document is the training of the classifier on this dataset. The subgroup discovery task searches for best conjunctive patterns of attribute value pairs for a given target attribute and returns a list of subgroups rated by the quality function. This procedure is performed for all target attributes in the dataset, which represent the cross product of the utilized types of descriptions and set of investigated entities. For each type of description and kind of boundary, only the subgroup with the highest score is utilized. This subgroup can be interpreted as a binary classification rule and described the consistent properties concerning the description and entity. Additionally, a threshold can be specified in order to accept only subgroups of a certain quality. It is sometimes better to provide no description of the consistencies than to introduce additional errors by poor analysis of the consistencies. The usage of a threshold depends on the configuration of the subgroup discovery tasks and on the domain. Improved quality functions like F_1^{exp} potentially induce subgroups with a lower quality score, but their description are often more useful when the input data is of poor quality. The result of the learning step is a set of subgroup that can be utilized for classifying consistent positions in the document.

6.1.1.6 Apply Classifiers on Dataset

The set of rule-based classifiers are applied on the given dataset in order to provide information about the consistent and inconsistent entities in the current document. Each rule classifies the set of examples, which represent the sequence of token in the document. Positive classified tokens indicate that this position fulfills the description for a consistent composition of that kind of entity. The actual information is given by the evaluation results of the rules compared to the labels in the dataset:

True Positive A true positive indicates that the rule-based description confirms with the predicted entity of the generated dataset. The aspect of the entity's composition is consistent compared to the other entities. This information does not introduce new knowledge about the position, but it can be utilized to increase the confidence that the entity is correctly identified.

False Positive A false positives indicates that the rule classified a token as a consistent position for the respective type of description, but no entity with this property is given in the predicted dataset. This error provides hints that an entity is missing or that the span of an entity is wrong. The confidence that the aspect of the entity's composition is fulfilled at this position can be increased for further information extraction models.

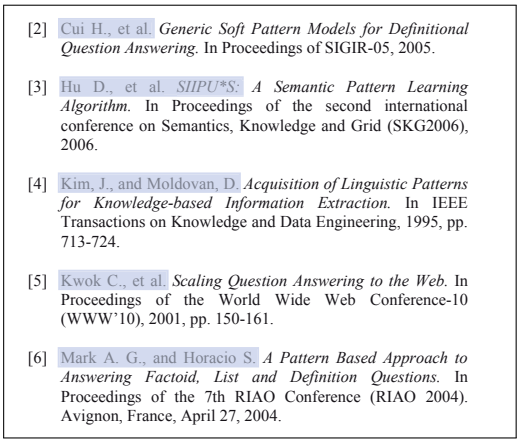
False Negative A false negative indicates that the given entity at this position does not confirm with the consistencies in the document. This error points out that either an additional but wrong entity was predicted or that the span of the entity is erroneous. The overall information provided by this error consists in the fact that there is no evidence that an entity should be present at this position concerning the aspect of its composition.

The union of the evaluation results of all rule-based classifiers and their positions provide a description of the consistent composition of the entities in the currently processed document.

The idea is that this knowledge can be exploited by further information extraction model in order to provide an improved identification of entities. The complete process of learning context-specific consistencies is illustrated with a running example for the segmentation of references in the next section.

6.1.2 Example

The process of learning context-specific consistencies is illustrated with an example for the segmentation of references. The subgroup discovery techniques are applied for learning a rule-based model for the consistent ends of the author entities. Other descriptions like the consistent begin of the author or consistent transitions between two kinds of entities can be learnt straightforwardly. The following examples sometimes refer to a token as a combination of several tokens for an increased visibility in the figures. The actual dataset would apply a common specification of tokens.


[2] Cui H., et al. *Generic Soft Pattern Models for Definitional Question Answering.* In Proceedings of SIGIR-05, 2005.

[3] Hu D., et al. *SIIPU*S: A Semantic Pattern Learning Algorithm.* In Proceedings of the second international conference on Semantics, Knowledge and Grid (SKG2006), 2006.

[4] Kim, J., and Moldovan, D. *Acquisition of Linguistic Patterns for Knowledge-based Information Extraction.* In IEEE Transactions on Knowledge and Data Engineering, 1995, pp. 713-724.

[5] Kwok C., et al. *Scaling Question Answering to the Web.* In Proceedings of the World Wide Web Conference-10 (WWW'10), 2001, pp. 150-161.

[6] Mark A. G., and Horacio S. *A Pattern Based Approach to Answering Factoid, List and Definition Questions.* In Proceedings of the 7th RIAO Conference (RIAO 2004). Avignon, France, April 27, 2004.

Figure 6.1: Excerpt of a reference section with highlighting for predicted author entities.

Figure 6.1 depicts the initial situation for the process. An arbitrary model provided predictions for the author entities. All of the entities but the second one have been classified correctly. The author in the second reference includes parts of the title. The text fragment "SIIPU*S" is falsely added to the author instead of the title.

This initial prediction is utilized to generate a training data for the rule learning algorithm, which is depicted in Figure 6.2. A training example is created for each token whereas only the last token of the author is determined as a true positive and the remaining tokens are specified as true negative. The examples are enriched with the features and properties of the surrounding tokens, which serve as boolean attributes for the rule learning algorithm. The value of the attribute is set to true if the feature or property occurs, and is set to false otherwise. The first true positive instance provides, for example, information about the period of the current token and the italic font of the following token. Then, subgroups are searched in this training dataset in a supervised fashion. The algorithm tries to identify the best combination of attributes concerning the quality function for classifying the true positive end of authors. A realistic result is the subgroup that

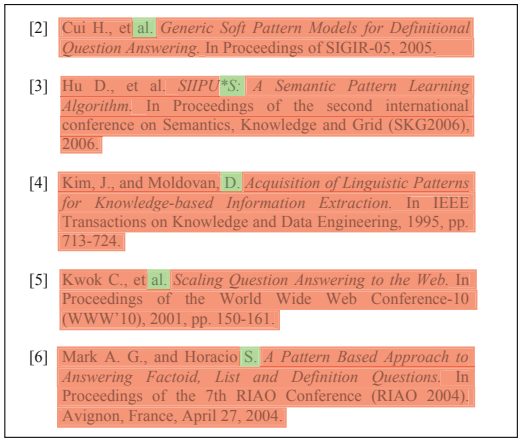


Figure 6.2: Generated training dataset for the rule learning algorithm. The last token of the author is determined as true positive and all remaining tokens are specified as true negative.

combines two attributes: the end of the author is located at a period that is followed by an italic token. This subgroup does not cover all true positive examples as it was not possible to find a suitable description that included the second end of author due to the restriction of conjunctions of binary attributes. Therefore, the score of the quality function for this subgroup is reduced.

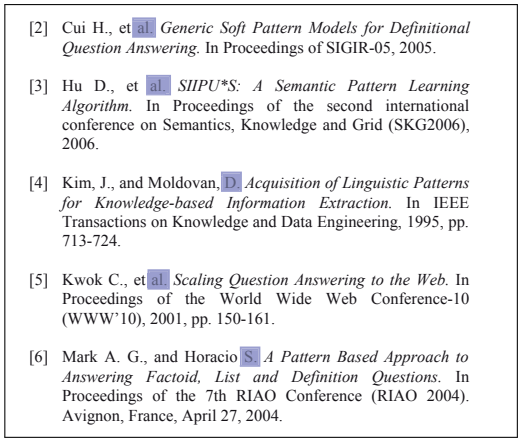


Figure 6.3: Result of classifying the end of the author using a rule in the form of "period followed by an italic token".

The subgroup can be interpreted a rule: if a period is followed by an italic token, then it is the last token of an author. The rule learnt on the generated training examples is then directly applied on the same examples. Figure 6.3 contains highlighting for the tokens, which have been classified by the rule as the end of the author. The rule was able to reproduce four ends of authors of the training dataset. The second reference contains a false positive ("al.") and

one false negative ("*S:") concerning the training examples. The false positive is, however, a correct end of author, which was erroneously labeled by the initial model. The rule learning algorithm was able to remove the errors and found the consistent ends of the author entity due to generalization. The evaluation of the rule on the training dataset can finally be utilized to create a more accurate labeling of the entities. The true positives indicate consistent boundaries, false negative point out incorrect boundaries, and false positives highlight missing boundaries. Including this information can greatly improve the performance of an information extraction model.

6.1.3 Experimental Results

The knowledge about the consistent composition of entities can be exploited in order to improve the accuracy of information extraction models. Section 6.2 and Section 6.3 will introduce and evaluate different approaches based on probabilistic graphical models that integrate this knowledge. This section directly evaluates the quality of learnt context-specific consistencies independently of the later usage. The experimental studies in this section provide insights on how well the classifiers can predict the correct composition of entities given erroneous data. This is achieved by comparing the accuracy of given entities to the accuracy of the classifier in the domains of references and curricula vitae.

The context-specific consistencies can be learnt with a vast amount of different configurations and methods. The evaluation of this section is restricted to a representative setting in the two domains concerning description of entities, considered kinds of entities and type of classifier.

Description of entities The entities are described by their boundaries, which provide a specification not depending on other types of entities and are easily exploited in graphical models. The boundaries of entities are overall a generic approach for describing consistencies and can be applied in many domains. The task of the classifier consists either in detecting the first token of a kind of entity or the last token of a kind of entity.

Type of classifier While almost all binary classifiers trained in a supervised fashion can be applied, the experimental studies only evaluate the subgroup discovery method. This method provides all important characteristics for the specific classification task and is applied for the evaluations in combination with the graphical models. Furthermore, it facilitates the usage of different quality functions. The experimental studies compare the common F_1 quality function with the optimized F_1^{exp} quality function.

Considered kinds of entities The domains considered in the experimental studies provide different kinds of entities. While all entities in curricula vitae are considered, the set of entities in references are restricted to the author, title and date. These entities can be assumed to be the most important entities of the domain and allow one to estimate the amount of their occurrence needed for the improved quality function.

The experimental evaluations in the following sections illustrate the power of subgroup discovery for learning context-specific consistencies in two scenarios. The first synthetic scenario in Section 6.1.3.1 investigates how well the learning process can handle erroneous input in general and provides a detailed analysis for the boundaries of the entities. In a second scenario

of Section 6.1.3.2, the classifiers are learnt on realistic data produced by a 5-fold Conditional Random Fields. Both scenarios utilize the datasets and features that are also applied in evaluation in Section 6.2 and Section 6.3. However, the evaluation of the classifiers was performed after the experimental study for Conditional Random Fields because it gives detailed insights in the datasets. This influences feature extraction or other configurations and may lead to optimizations for the datasets and domain. Furthermore, the experiments pointed out several labeling errors in the dataset, which have been fixed for this evaluation, but which are still present in the later studies.

6.1.3.1 Random Synthetic Errors

This experimental evaluation investigates the robustness of the subgroup discovery task to predict correct boundaries. The scenario is configured the following way: the complete dataset with correct entities is incrementally deteriorated by replacing a correct boundary by a randomly selected, erroneous position. This process is iterated until all entities possess no correct boundary resulting in a completely random and falsely annotated set of entities. The subgroup for describing either the begin or the end of an entity are learnt for each iteration and thus rely on increasingly defective input data. For investigating the applicability of the approach, separate evaluations are given for each kind of boundary and entity. An additional analysis of all combined boundaries illustrates the overall performance.

The results are given in Figure 6.4 to Figure 6.15. Each figure contains two scatter plots that display the F_1 score of the classification rules dependent on the F_1 score of the input data. Data points in diagrams for specific boundaries of entities depict the F_1 score for a complete document. In diagrams for all entities, a data point indicates the average F_1 score of all boundaries for the complete dataset. An accumulation of data points that lies above the imaginary diagonal represents an improvement compared to the input data. Data points located below the diagonal indicate that the subgroup was not able to model the consistencies and thus introduced additional errors. The left diagram in each figure displays the results of the subgroup discovery task using the common F_1 measure for quality function, whereas the right diagram contains the results using the improved F_1^{exp} measure.

References Figure 6.4 and Figure 6.5 contain the results for identifying the begin of an author and the end of the author respectively. Subgroups using the normal F_1 quality function are able to greatly improve the boundaries of the author. Almost all data point are located above the diagonal indicating that the learnt description was able to remove most errors. The accumulation of data points near an F_1 score of 0.0 occurs only for an F_1 score of about 0.3, which indicates the minimal requirements for a successful task. However even for this low quality input, there are many data points indicating a high score. The subgroups using the improved F_1^{exp} quality function achieve even better results. The accumulation of data points is clearly moved towards an F_1 score of 1. This fact can be especially observed in the left part of the diagrams.

The results for predicting the correct boundaries of the title are depicted in Figure 6.6 and Figure 6.7. The begin of the title is reliably identified using both quality functions since almost all data points are located above the diagonal. The results using the improved F_1^{exp} quality function are even better. Only the beginnings in a small amount of documents are not near the F_1 score of 1 regardless of the quality of the input. The end of the title causes more problems. Most of the

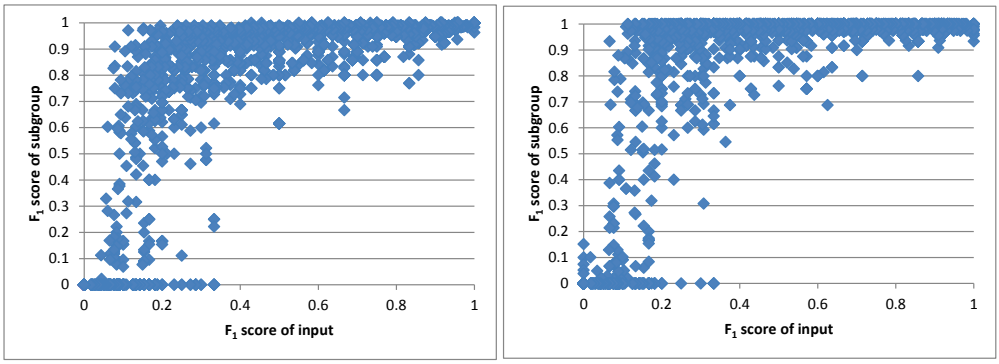


Figure 6.4: Classification of the **author's begin** in reference sections with subgroup discovery using F_1 (left) and F_1^{exp} (right) measure as quality function. A data point represents the F_1 score of one document.

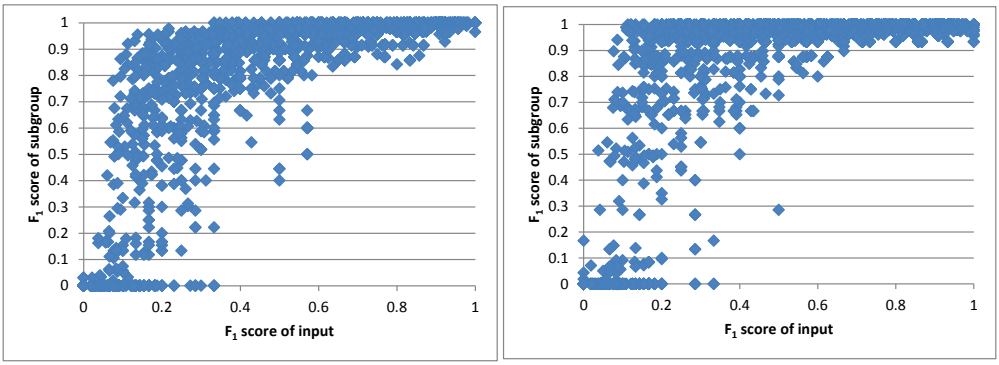


Figure 6.5: Classification of the **author's end** in reference sections with subgroup discovery using F_1 (left) and F_1^{exp} (right) measure as quality function. A data point represents the F_1 score of one document.

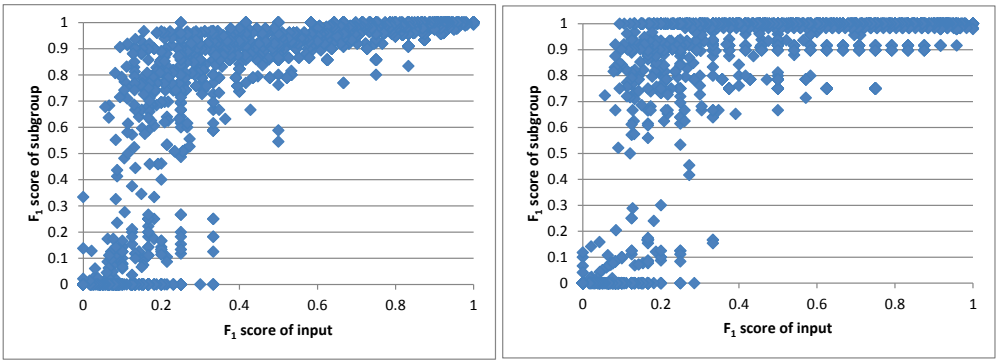


Figure 6.6: Classification of the **title's begin** in reference sections with subgroup discovery using F_1 (left) and F_1^{exp} (right) measure as quality function. A data point represents the F_1 score of one document.

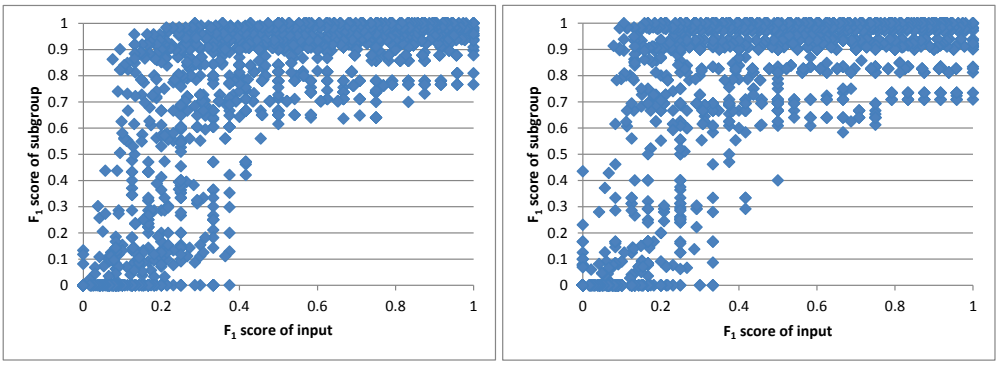


Figure 6.7: Classification of the **title's end** in reference sections with subgroup discovery using F_1 (left) and F_1^{exp} (right) measure as quality function. A data point represents the F_1 score of one document.

data points are indeed located above the diagonal, but the ending of the title cannot always be modelled using the available features. This fact can be observed at the data points in the right part of the diagram. In some documents, the F_1 score of the subgroup is lower than the F_1^{exp} score of the input, which can be observed in the right part of the diagrams where data points are located near an F_1 score of 0.7. The end of the title is sometimes hard to distinguish from the end of booktitles.

Figure 6.8 and Figure 6.9 contain the results for identifying the beginning and end of a date. The subgroups are able to detect most boundaries independently of the errors in the input document. While the results of the normal quality function are already very good, the subgroups using the improved quality function still provide better results. The data points of the right diagrams are generally located nearer to an F_1 score of 1.0. The improved quality function introduces a few errors for the beginning of a date, which can be observed by the single data points near an F_1 score of 0.8. This problem is caused by the fact that the assumption about the

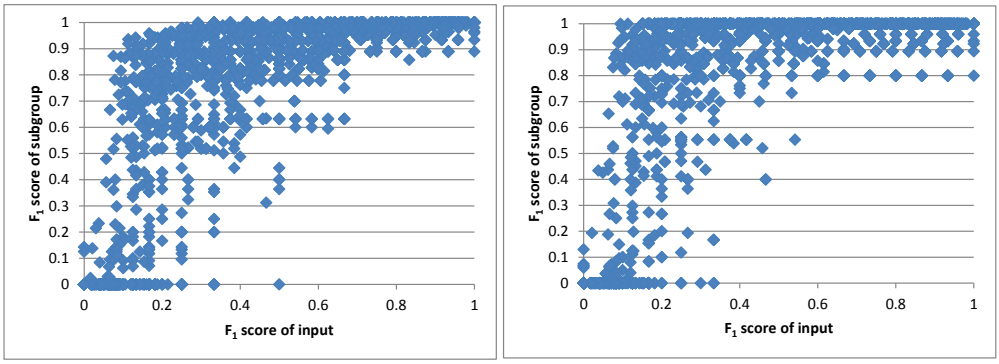


Figure 6.8: Classification of the **date's begin** in reference sections with subgroup discovery using F_1 (left) and F_1^{exp} (right) measure as quality function. A data point represents the F_1 score of one document.

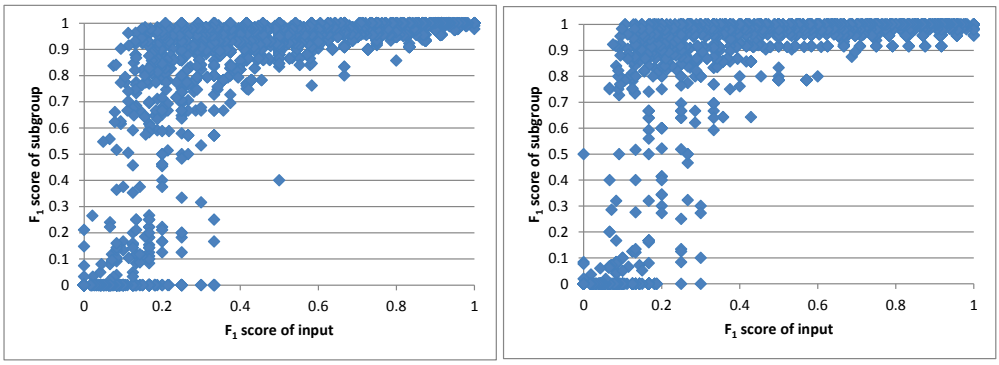


Figure 6.9: Classification of the **date's end** in reference sections with subgroup discovery using F_1 (left) and F_1^{exp} (right) measure as quality function. A data point represents the F_1 score of one document.

amount of entities used in the improved quality function does not hold. Hence, the subgroup provides a description that includes additional beginnings. This problem does not occur for the end of the date, which is overall easier to detect than the beginning. Especially using the improved quality function, only a few data points are located in the middle of the diagram. The subgroups are able to identify the end in most documents with an F_1 score of at least 0.9. If the quality of the input drops below an F_1 score of 0.2, the classifier are not able to learn the correct boundaries.

Figure 6.10 contains an overview of results for all boundaries. In this diagram, a data point represents the average F_1 score for all boundaries in the complete dataset dependent on the quality of the input. Both curves indicate that the approach is able to greatly improve the identification of the boundaries because the data point are located above the imaginary diagonal. The subgroups using the common F_1 quality function achieve an average F_1 score of 0.9 already for an input of an F_1 score of 0.4. The subgroups using the improved F_1^{exp} quality function

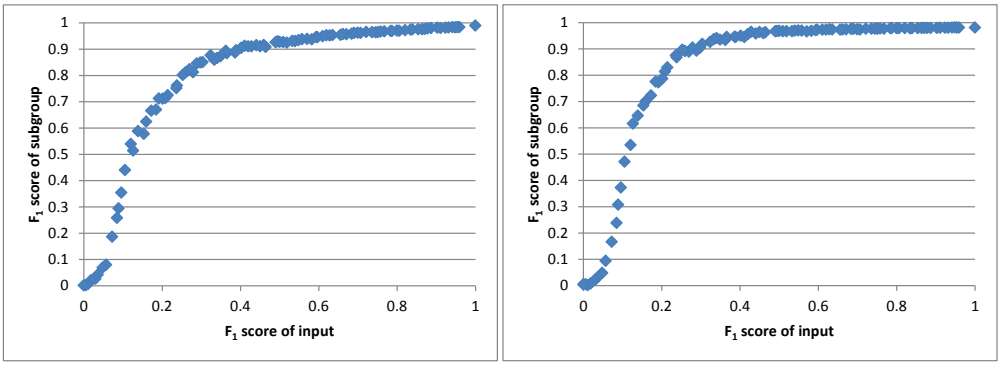


Figure 6.10: Classification of boundaries of the author, title and date in reference sections with subgroup discovery using F_1 (left) and F_1^{exp} (right) measure as quality function. A data point represents the average F_1 score for the complete dataset.

perform even better. Here, an F_1 score of 0.9 is already reached for an input of an F_1 score of 0.3. It can be observed that the second curve for the improved quality function generally climbs steeper and approaches the F_1 score of 1.0 faster. This property can especially be noted near an F_1 score of 0.6 for the input quality.

Curricula Vitae Figure 6.11 and Figure 6.12 contain the results for identifying the beginning and end of the client entities that refer to the company or sector, in which the author of the document was employed. A first look at the diagrams reveals that consistencies in this domain are harder to model by subgroup discovery. Most of the predicted boundaries have been improved compared to the given input data since most of the data points are located above the imaginary diagonal. A closer look at the beginning of the client indicates that both quality functions are able to provide good results. The improved quality function produces overall more concise prediction of the beginning of the client, but introduces also some additional errors located at the lower right part of the second diagram. These errors are caused by a combination of the available features and the assumption about the amount of occurrences of the entity. The subgroups are not able to describe consistent composition for the expected amount of beginning. The beginning of the client is hard to identify in some documents, even for humans.

The end of the client also provides valuable results whereas the improved quality function F_1^{exp} performs much better than the normal quality function F_1. This improvement can especially be observed by the accumulation of data points near the F_1 score of 1.0. In both diagrams, a line of data points near the F_1 score 0.7 can be identified. The subgroups have not been able to provide a correct description for some documents independently of the input's quality. The entities in the original documents have been consistently structured, but the conversion into plain text introduced some conflicts, e.g., by moving a few clients from the second line to the third line of a project. The subgroup relied, however, on the number of line for the end of the client since no better features were available.

The results for predicting the correct boundaries of the date are depicted in Figure 6.13 and Figure 6.14. The results of the subgroups using the normal F_1 quality function show a

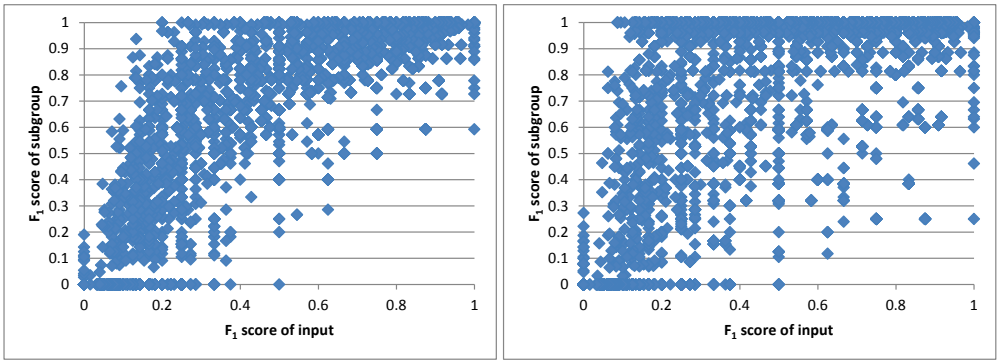


Figure 6.11: Classification of the **client's begin** in curricula vitae in reference sections with subgroup discovery using F_1 (left) and F_1^{exp} (right) measure as quality function. A data point represents the F_1 score of one document.

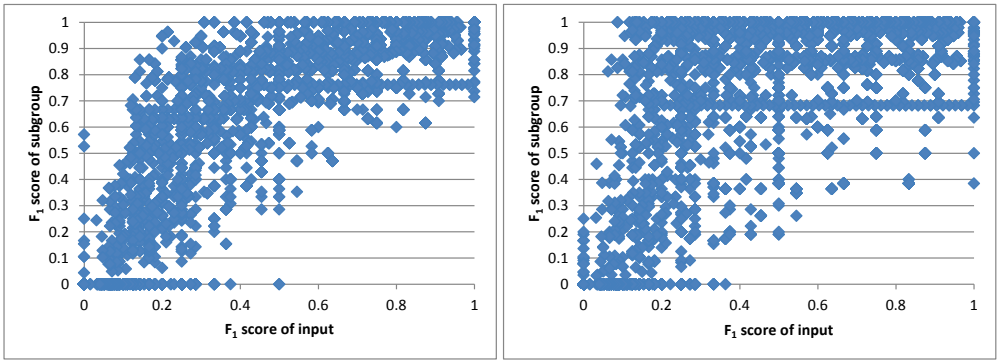


Figure 6.12: Classification of the **client's end** in curricula vitae with subgroup discovery using F_1 (left) and F_1^{exp} (right) measure as quality function. A data point represents the F_1 score of one document.

clear accumulation of data point above the imaginary diagonal. Only a few outliers indicate additional errors introduced by the subgroups. The improved quality function F_1^{exp} achieves again better results than the normal one. This improvement can be observed by the sparse data point in the middle of the diagrams. The diagrams for the beginning of the date show that theassumption about the amount of occurrences does not hold for some documents. Here, no suitable descriptions were found that fulfill the expectation. As for the end of the date, modeling of the properties is even harder indicated by the accumulation of data point below an F_1 score of 1.0 for an input quality of an F_1 score of 1.0.

Figure 6.15 contains an overview of results for all boundaries where one data point reflects the average F_1 score for the complete dataset dependent on the quality of the input. The subgroups using the improved quality function F_1^{exp} overall performs better. This fact can be observed by the steeper climb and earlier approach of the F_1 score of 1.0. However, both quality functions are not able to reach an F_1 score of 1.0 since either the features are not sufficient to model the boundaries or the entities are not consistently structured. The subgroups using the F_1^{exp}

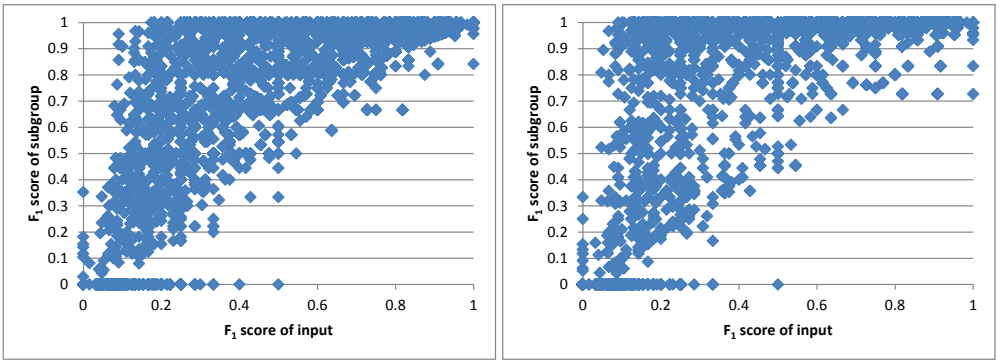


Figure 6.13: Classification of the **date's begin** in curricula vitae with subgroup discovery using F_1 (left) and F_1^{exp} (right) measure as quality function. A data point represents the F_1 score of one document.

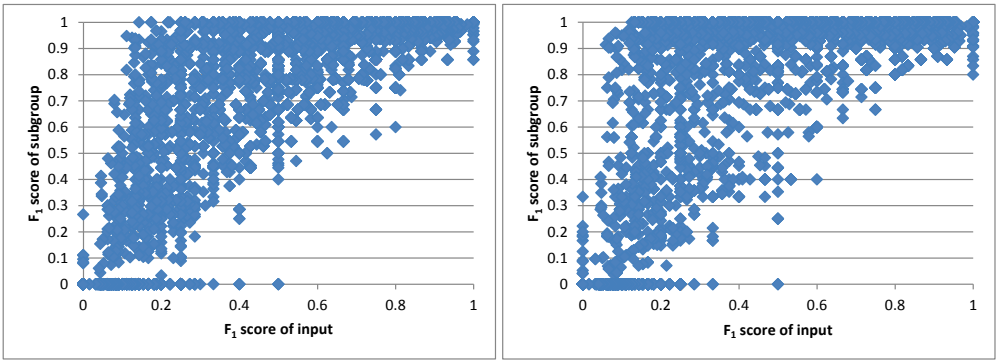


Figure 6.14: Classification of the **date's end** in curricula vitae with subgroup discovery using F_1 (left) and F_1^{exp} (right) measure as quality function. A data point represents the F_1 score of one document.

quality function achieve an average F_1 score of 0.9 already at an input quality of 0.5 whereas the subgroups using the F_1 quality function reaches this prediction yet with a input quality greater than 0.6.

Summary Subgroup discovery performs well for predicting the consistent boundaries in both domains. The experimental setting introduces a false positive and a false negative in each iteration by replacing a true positive. This additional deterioration of the input data is however not as problematic since the introduced false positive boundaries are chosen randomly. The correct composition of the entities can still be induced by the subgroups because the no valid descriptions can be learnt that cover the additional errors. It can be assumed that the subgroups perform equally well if no false positives are included in the input data. The subgroups using the improved quality function F_1^{exp} achieve better results for all boundaries in both datasets if the assumption about the expected amount of entities holds. While the F_1 measure tries to reproduce the true positives and the false positives in the input data, the F_1^{exp} measure additionally accounts

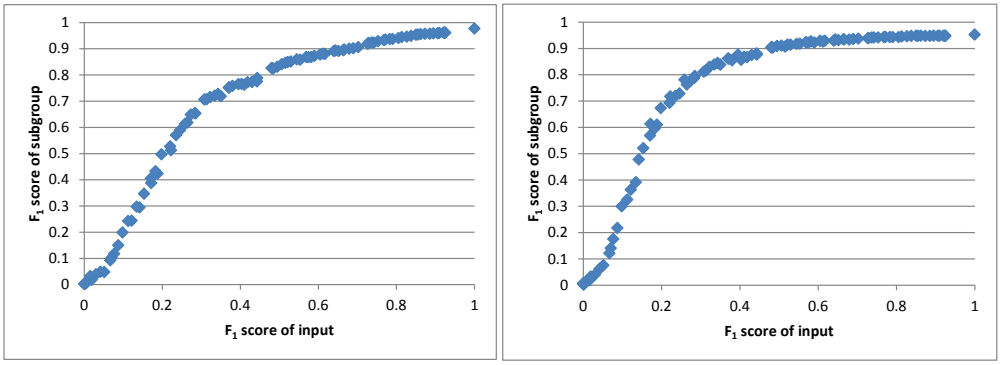


Figure 6.15: Classification of boundaries of the client and date in curricula vitae with subgroup discovery using F_1 (left) and F_1^{exp} (right) measure as quality function. A data point represents the average F_1 score for the complete dataset.

the amount of covered boundaries. Hence, it is able to learn the correct properties of a boundary even if only a small amount of true positives is included in the input data.

This setting provides valuable insights in the advantages of the approach, but does not provide a realistic scenario. A normal model for information extraction produces a different kind of input. The predictions of the entities include rather systematic errors than random ones. This aggravates the induction of correct subgroups since the dominant composition of the predicted entities may actually be incorrect, which leads to a completely misleading subgroups. Furthermore, a realistic information extraction model often identifies all entities correctly in one document but it is not able to find any correct entities in another document. While learning context-specific consistencies can only deteriorate the result in the first document, it is not able to produce any results in the second document. The approach works only if a reasonably correct prediction is provided.

6.1.3.2 Realistic Prediction

The discussion in the last section pointed out the limited expressiveness of the evaluation based on random errors. This section investigates the capabilities of subgroup discovery for learning consistencies on realistic predictions. For this purpose, a Conditional Random Field is utilized in a 5-fold fashion. The model is trained on four fifths of the dataset and then applied on the remaining fifth part. This process is iterated for all permutation in order to accomplish a labeling of the entities that does not suffer from overfitting. The boundaries predicted by the Conditional Random Field are then used as the input of the subgroup discovery task. In this experimental setting, only the improved quality function F_1^{exp} is applied. A disadvantage of this setting is the coverage of different qualities of the input data. The models are hardly configurable to produce a wide variation of F_1 scores. Hence, the evaluation is limited to a realistic scenario where the Conditional Random Field is trained in a optimal setting with enough iterations.

Table 6.1 contains the results of the experimental study. The Conditional Random Field is able to achieve an average F_1 score of 0.960 for the references dataset and an average F_1 score of 0.824 for the curricula vitae dataset. Both scores represent very good and realistic results for these

	CRF	$SG_{F_1^{exp}}$
References	0.960	0.973
Curricula Vitae	0.824	0.852

Table 6.1: Average F_1 scores for the boundaries of entities in two domains. The F_1 scores of the CRF have been produced in a 5-fold evaluation. The subgroups utilized the F_1^{exp} quality function and are induced on the result of the CRF.

domains and datasets. The results refer always to the evaluation of the boundaries alone. Since the boundaries are however generated using the predicted entities, the score are hardly affected. The subgroups achieved an average F_1 score of 0.973 and 0.852 when processing the output of the Conditional Random Fields. This indicates and error reduction of about 30% for references and about 16% for curricula vitae. The results provide only an indication of possible improvements for a model that exploits the learnt consistencies. Even if the absolute improvement is not perfect, the learnt boundaries may enable another model to achieve proportionally better results. If the subgroups predicted several false positive boundaries additionally to the correct ones, then an information extraction model can still exploit this information for extracting the correct entities.

6.2 Stacked Conditional Random Fields

This section introduces a method for exploiting context-specific consistencies by combining two linear-chain Conditional Random Fields (CRFs) in a stacked learning framework. After the instances are initially labeled, a rule learning method is applied on label transitions within one context in order to identify their shared properties. The stacked CRF is then supplemented with high-quality features that help to resolve possible ambiguities in the data. While the features possess a global meaning for the model, their manifestation adapts to the patterns occurring in the currently processed document. The approach is evaluated with a real-world dataset for the segmentation of references, a domain that is widely used to assess the performance of information extraction techniques. The results show a significant reduction of the labeling error and confirm the benefit of additional features induced online during processing the data. This approach builds upon earlier work on stacked CRFs that relies on mutual information in order to analyze the consistencies [194][49].

[49] The contents of this section have been published in two research papers: Peter Kluegl, Martin Toepfer, Florian Lemmerich, Andreas Hotho, and Frank Puppe. Stacked Conditional Random Fields Exploiting Structural Consistencies. In Pedro Latorre Carmona, J. Salvador Sanchez, and Ana Fred, editors, Proceedings of 1st International Conference on Pattern Recognition Applications and Methods (ICPRAM), pages 240–248, Vilamoura, Algarve, Portugal, February 2012. SciTePress [128]. Peter Kluegl, Martin Toepfer, Florian Lemmerich, Andreas Hotho, and Frank Puppe. Exploiting Structural Consistencies with Stacked Conditional Random Fields. Mathematical Methodologies in Pattern Recognition and Machine Learning. Springer New York, 2013. 111-125 [129].

6.2.1 Stacked Inference with Consistencies

Sequence labeling methods like CRFs assign a sequence of labels $\mathbf{y} = (y_1, \ldots, y_T)$ to a given sequence of observed tokens $\mathbf{x} = (x_1, \ldots, x_T)$. Let $crf(\mathbf{x}, \Lambda, F) = \mathbf{y}$ be the function that applies the CRF model with the weights $\Lambda = \{\lambda_1, \ldots, \lambda_K\}$ and the set of feature functions $F = \{f_1, \ldots, f_K\}$ on the input sequence $\mathbf{x}$ and returns the labeling result $\mathbf{y}$. The set of model weights must of course correspond to the set of feature functions. Since the CRF processes this sequence of tokens in one labeling task, $\mathbf{x}$ is called an instance. All instances together form the dataset D which is split in a disjoint training and testing subset. An information or entity consists often of several tokens and are encoded by a sequence of equal labels. It is assumed here that the given labels already specify an unambiguous encoding. An instance itself may contain multiple entities specified by an arbitrary amount of labels, one label for each token of the input sequences. Furthermore, it is assumed that the dataset $D = \{C_1, \ldots, C_n\}$ can be completely and disjointly partitioned into subsets of instances $\mathbf{x}$ that originate from the same creation context C_i. Similar to the relational template in [131], this work implies that a trivial context template exists for the assignment of the context set.

In stacked graphical learning, several models can be stacked in a sequence. Experimental results, e.g., of Kou [131], have shown that this approach already converges with a depth of two learners and no significant improvements are achieved with more iterations of stacking. Therefore, stacked graphical learning with CRFs is only applied in a two-stage approach like Krishnan and Manning [134]. In order to extract entities collectively, this work defines the stacked inference task on the complete set of instances $\mathbf{x}$ in one context C. The two CRFs, however, label the single instances within that context separately as usual. The following algorithm summarizes the stacked inference combined with online rule learning. Afterwards, integration of the rule learning techniques for the identification of context-specific consistencies is described. Details about the estimation of the weights (e.g., Λ^m) are discussed in Section 6.2.2.

1. **Apply base CRF** Apply $crf(\mathbf{x}, \Lambda, F) = \hat{\mathbf{y}}$ on all instances $\mathbf{x} \in C$ in order to create the initial label sequences $\hat{\mathbf{y}}$.

2. **Learn consistencies** Learn classification rules for the target attributes $\hat{\mathbf{y}}$ and construct a feature function $f^m \in F^m$ for each discovered rule.

3. **Apply stacked CRF** Apply $crf(\mathbf{x}, \Lambda \cup \Lambda^m, F \cup F^m) = \mathbf{y}$ again on all instances $\mathbf{x} \in C$ in order to create the final label sequences $\mathbf{y}$.

This process is illustrated in Figure 6.16. The dataset consists of a list of instances. Some of them origin from the same context, for example, the first four and the last four instances. These instances are individually processed by the base CRF and the predicted label sequence is applied together with the features for learning a rule-based model about the context-specific consistencies. The resulting rules specify the semantics of the additional features in the stacked model, which hopefully creates a more accurate prediction.

Rule learning is applied on all (probably erroneously) label assignments $\hat{\mathbf{y}} \in C$ of the base CRF. The rules are learnt in order to classify certain label transitions and, thus, describe the shared properties of the transition within the context C. The labeling error in the input data is usually eliminated by the generalization of the rule learning algorithm. The label transition is optimally described by a single pattern that covers the majority of transitions despite of

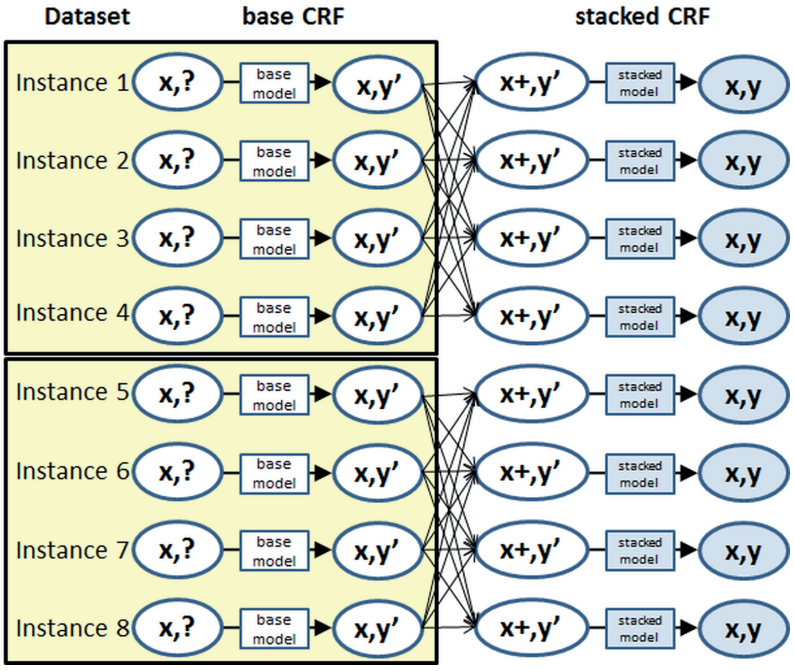


Figure 6.16: Overview of the stacked inference.

erroneously outliers. The learnt rules are then used as binary feature functions in the same context C: they return 1 if the rule applies on the observed token x_t, and 0 otherwise. The model gains therefore additional features that indicate label transitions if the instances are consistently structured. Even if the learnt rules are misleading due to erroneously input data or missing consistency of the instances, their discriminative impact on the inference is yet weighted by the learning algorithm of the stacked CRF.

This process is illustrated by a simple example concerning the author label, but can also be applied to any other label. Let a reference section be processed by the base CRF that classified all instances but one correctly. For some reasons the base CRF missed the date and editor and misclassified the tokens x_5 to x_{10} and the tokens x_{18} to x_{43} in the ninth reference (cf. Figure 6.17). The input of the rule learning now consists of 22 transitions from author to date whereas one transition is incorrect. In this case, a reasonable result of the rule learning is the rule "if the token x_t is a period and followed by a parenthesis, then there is a transition from author to date at t". Converted to a feature function, this rule returns 1 for token x_4 and 0 for all other tokens of the reference in Figure 6.17. The weight of this new feature function is then estimated by the stacked CRF. Therefore, the stacked CRFs' likelihood of a transition from author to date is increased at the token x_4 and decreased at the token x_{41} due to the presence or absence of the meta-features. Finally, the set of learnt rules are transformed to the set of binary feature functions F^m that return true, if the condition of the respective rule applies.

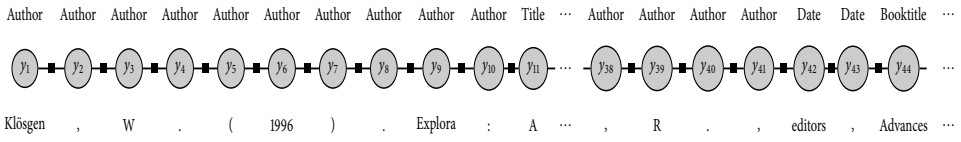


Figure 6.17: Two excerpts of a reference with erroneous labeling: The date (y_5 to y_8) and the begin of the title (y_9 and y_{10}) was falsely labeled as author, e.g., due to the high weight of the colon for the end of an author. The editor was additionally labeled as an author (up to y_{41}) and date (y_{42} and y_{43}).

6.2.2 Parameter Estimation

The weights of two models need to be estimated for the presented approach: the parameters of the base model and of the stacked model. The base model needs to be applied on the training instances for the estimation of the weights of the stacked model, i.e., step 1 and step 2 of the stacked inference in Section 6.2.1 need to be performed on the training set. If the weights of the base model are estimated as usual using the labeled training instances, then it produces unrealistic prediction on these instances and the meta-features of the stacked model are overfitted resulting in a decrease of accuracy. Since the base model is optimized in this case on the training instances, it labels these instances perfectly. The learnt rules create optimal descriptions of the structural consistencies and the stacked model assigns biased weights to the meta-features. This is of course not reproducible when processing unseen data. It can reduce the accuracy of the stacked inference, even if the meta-features in the testing phase are also overall of good quality.

The simple solution to this problem is a cross-fold training of the base model for training of the stacked CRF that was already successfully applied by several approaches [131, 134]. Training of the base model in a cross-fold fashion is also a very good solution for the presented approach, but this work simply decreases the accuracy of the model by reducing the training iterations. Thus, only one model needs to be trained for the learning phase of the stacked model. For the testing phase or common application however, a single base model learnt with the default settings is applied.

The model of the stacked CRF is trained dependent on the base model and the creation context C that are both applied to induce the new features online during the stacked inference. The weights $\Lambda = \{\lambda_1, \ldots, \lambda_K\}$ and $\Lambda^m = \{\lambda_1^m, \ldots, \lambda_M^m\}$ of the stacked CRF are estimated to maximize the conditional probability on the instances of the training dataset:

$$P_\lambda\Big(\mathbf{y}|\mathbf{x}, C, crf(\mathbf{x}, \Lambda', F)\Big) = \frac{1}{Z_\mathbf{x}} \exp\left(\sum_{t=1}^{T}\sum_{i=1}^{K} \lambda_i \cdot f_i(y_{t-1}, y_t, \mathbf{x}, t) + \sum_{t=1}^{T}\sum_{j=1}^{M} \lambda_j^m \cdot f_j^m\Big(y_{t-1}, y_t, \mathbf{x}, t, C, crf(\mathbf{x}, \Lambda', F)\Big)\right) \quad (6.5)$$

The resulting model still relies on the normal feature functions but is extended with dynamic and high quality features that help to resolve ambiguities and substitute for other missing features. These meta-features possess the same meaning in the complete dataset, but change

their interpretation or manifestation dependent on the currently processed creation context. They provide overall a very good description of the structural consistencies and are often alone sufficient for a classification of the entities.

A short example: The induced feature function for the transition of the author to the date is set to very high weights for the corresponding state transition of the learnt model. This feature function returns 1 in the exemplary reference section for a token, which is a period and is followed by a parenthesis. In other reference sections with a different style guide applied, the feature function for this state transition returns 1, if the token is a colon and is followed by a capitalized word. However, both examples refer only to exactly one feature function that dynamically adapts to the currently processed context.

6.2.3 Experimental Results

The presented approach is evaluated in the domain of reference segmentation (cf Section 3.2.1). The common approach is to separately process the instances, namely the references. Within these references, the interesting entities need to be identified. Since all tokens of a reference are part of exactly one entity, one speaks of a segmentation task. This section introduces the overall settings and presents the experimental results.

6.2.3.1 Datasets

All available and commonly used datasets for the segmentation of references are a listing of references without their creation context and are thus not applicable for the evaluation of the presented approach. Therefore, a new dataset was manually annotated with the label set of Peng and McCallum [160] concerning the fields *Author*, *Booktitle*, *Date*, *Editor*, *Institution*, *Journal*, *Location*, *Note*, *Pages*, *Publisher*, *Tech*, *Title* and *Volume*. The resulting dataset contains 566 references in 23 documents extracted only of complete reference sections of real publications. The amount of instances is comparable to previously published evaluations in this domain, cf., [160, 52].

Similar to previous studies with CRFs in the domain of reference segmentation, this work uses features indicating the length of tokens, the relative position inside the reference string, n-gram prefixes, n-gram suffixes, as well as the covered text of the token and the covered text of tokens on the left and on the right of the observed token. Additionally, features representing token classes, dictionaries, regular expressions for URLs and simple combinations of features, for example a first name followed by a capitalized word, are integrated. The dictionaries cover first names, locations, keywords (e.g., "eds.") and some well-known journals and publishers. Only this additional subset of features is used for the rule learning task. This restriction is justified with the minimal expressiveness of ngram features for the identification of the structure in relation to the increase of the search space. Overall, the applied features are comparable to previously published approaches, e.g., [160, 52]. Only a part of the basic features is used for the induction of the meta-features, omitting ngram and token window features. This restriction is justified with their minimal expressiveness for the identification of the consistencies in relation to the increase of the search space.

CRF	A single CRF trained on the same data and features.
STACKED CRF	A two-stage CRF approach. The predictions of the base CRF are added as features to the stacked CRF.
STACKED+DESCRIPTIVE	The default approach of stacked CRF combined with subgroup discovery for rule learning. Only transitions between the labels *Author*, *Title*, *Date* and *Pages* that commonly occur in most references are considered.
STACKED+MORE	A stacked approach using subgroup discovery that additionally learns the transitions of the labels *Booktitle*, *Journal* and *Volume*.
STACKED+MAX	A stacked approach using subgroup discovery that considers the transitions of all labels for the rule learning task.

Table 6.2: Overview of the evaluated models.

6.2.3.2 Implementation Details

The machine learning toolkit Mallet[50] is used for an implementation of the CRF in the presented approach. For rule learning, this work chose a subgroup discovery implementation[51] because of the multifaceted configuration options that allow a deep study of the approach's limits. Only the default parameters are used for the CRF and all evaluated models were trained until convergence. Only for the training of the stacked model, the iterations of the base model was reduced to 50 iterations. For the default configuration of both rule learning tasks, the window size is set to $w = 1$. Additionally for the default setting of the subgroup discovery, this work uses a quality function based on the F_1 measure, selected only one rule for each description of a label, restricted the length of the rules to maximal three selectors, and set an overall minimum threshold of the quality of a rule equal to 0.5. No improved quality function was applied since assumptions about the occurrence of the entities do not hold for all kind of entities. The presented approach is overall straightforward to implement and only established standard methods are used. Its inference is still efficient in contrast to complex models with approximate inference techniques.

6.2.3.3 Results

The presented approach is compared to two base line models in a five-fold cross evaluation. Three different settings of stacked CRFs combined with a rule learning technique are investigated. A detailed description of all evaluated models is given in Table 6.2. The documents of the dataset are randomly distributed over the five folds. The results of the experimental study are depicted in Table 6.3. Only marginal differences can be observed between the two base line models CRF and STACKED CRF. This indicates that the normal stacking approach cannot exploit the structural consistencies or gain much advantage of the predicted labels.

[50] `http://mallet.cs.umass.edu`
[51] `http://sourceforge.net/projects/vikamine/`

	F_1
Base Line	
CRF	0.913
STACKED CRF	0.918
Presented Approach	
STACKED+DESCRIPTIVE	0.940
Variants of DESCRIPTIVE	
STACKED+MORE	0.941
STACKED+MAX	0.936

Table 6.3: F_1 scores averaged over the five folds.

All of the stacked models combined with rule learning techniques significantly outperform the base line models using a one-sided, paired t-tests on the F_1 scores of the single references ($p \ll 0.01$). Comparing the results of STACKED+DESCRIPTIVE that only considers the consistencies of four labels to the base line CRF, the approach achieves an average error reduction of over 30% on the real-world dataset.

The second configuration with a subgroup discovering technique STACKED+MORE considers the transition between seven labels and is able to slightly increase the measured F_1 score compared to the default model STACKED+DESCRIPTIVE. STACKED+MAX that induces rules for all labels achieves only an average error reduction of 26% compared to a single CRF. This is mainly caused by misleading meta-features for rare labels. The task of learning consistencies from a minimal amount of examples is error-prone and can decrease the accuracy, especially if the examples are labeled incorrectly.

Table 6.4 provides closer insights in the benefit of the presented approach using the author label as an example. STACKED+DESCRIPTIVE is able is significantly improve the labeling accuracy for all folds but one. The third fold contains an unfavorable distribution of style guides between the training and testing set for the author. If the initial base CRF labels a label systematically incorrectly, then the rule learning cannot induce any valuable and correct descriptions of the structure. Nevertheless, an average error reduction of over 50% is achieved for identifying the author of the reference.

For comparison, the skip-chain approach of [189] has been applied with factors for capitalized words and additionally for identical punctuation marks, but no improvement over the base line models could be measured. Furthermore, the feature induction for CRFs [150] was integrated, but resulted counter-intuitively in a decrease of the accuracy.

The performance time of the presented approach for one fold averaged over the five folds is about nine times faster than a higher-order model with skip edges. The difference in speed is less compared to previously published evaluations [131]. This is mainly caused by the fact that the rule learning is neither optimized for this task nor for the domain, e.g., by pruning the attributes.

	CRF	Stacked+ Descriptive	error reduction
Fold 1	0.977	0.996	82.6%
Fold 2	0.970	0.992	73.3%
Fold 3	0.964	0.965	2.8%
Fold 4	0.971	0.988	58.6%
Fold 5	0.895	0.951	53.3%
average	0.955	0.978	51.6%

Table 6.4: F_1 scores of the author label.

6.3 Towards Higher-order Models

The approaches of this section investigate how the context-specific consistencies can be exploited with the idea of skip-chain CRFs or in general CRFs with additional potentials for long-range dependencies. In contrast to skip-chain CRFs, where the potentials are only based on the token sequence (cf. Equation 2.11), the additional potentials are mainly dependent on the label sequence.

The approaches need a prediction of the assignment in order to be able to link or relate the entities. While there are a variety of different ways to model the consistencies, the presented approaches in this section investigate only the boundaries of the entities in order to describe their consistent composition. The label sequence (hidden variables) is of course not available during inference when the graph is unrolled on an instance with all potentials since it is the result of the computation of $p_\theta(\mathbf{y}|\mathbf{x})$. However, there are many different ways to provide a prediction of the label sequence during inference. The initial choice was to incrementally unroll the graph: First, the potentials of the linear-chain part are unrolled, the currently most likely label sequence is computed and this prediction is used to further unroll the additional potentials. However, this work observed problems with the parameter estimation and inference mechanism (cf. Section 6.3.3). While the model sometimes achieved remarkable improvements, the approach frequently did not converge at all. Therefore, this work utilizes a separate static linear-chain model in order to provide a constant prediction of the label sequences, which corresponds to the approach of stacked graphical models [131, 134, 128] (cf. Section 6.2). Here, an initial model is used to compute new features for a stacked model. In the approaches of this section, however, the predicted assignments of the initial model lead to additional potentials. Normally, cross-fold training is applied for the initial model in order to prevent unrealistic predictions during training of the stacked model. This improvement is neglected in the belief that the advantages of the presented models prevail[52].

[52] The contents of this section have been published in Peter Kluegl, Martin Toepfer, Florian Lemmerich, Andreas Hotho, and Frank Puppe. Collective information extraction with context-specific consistencies. In Peter A. Flach, Tijl De Bie, and Nello Cristianini, editors, ECML/PKDD (1), volume 7523 of Lecture Notes in Computer Science, pages 728–743. Springer, 2012 [127].

6.3.1 Comb-chain CRFs

In a first approach, the variables of a linear-chain model are extended with additional (unigram) factors dependent on the classification result (cf. Figure 6.18). Hence, this work chose the name comb-chain CRFs for this approach because of the layout of the graph.

Let $\mathcal{R}_b(y)$ and $\mathcal{R}_e(y)$ be the set of positions, which are identified by the classifier as the beginning and end of an entity with the label y. The positions of additional factors are defined the following way:

$$\begin{aligned} \mathcal{U}_b &= \left\{ u : y_{u-1} \neq y_u \vee u \in \bigcup_y \mathcal{R}_b(y) \right\} \\ \mathcal{U}_e &= \left\{ u : y_u \neq y_{u+1} \vee u \in \bigcup_y \mathcal{R}_e(y) \right\} \\ \mathcal{U} &= \mathcal{U}_b \cup \mathcal{U}_e \end{aligned} \tag{6.6}$$

$\mathcal{U}_b$ and $\mathcal{U}_e$ contain all positions that are either intermediately labeled by the external model or are identified by the classifier as the beginning, respectively end of an entity. The conditional probability is then defined as[53]

$$p_\theta(\mathbf{y}|\mathbf{x}) = \frac{1}{Z(\mathbf{x})} \prod_{t=1}^{T} \Phi_{\text{L}}(y_t, y_{t-1}, \mathbf{x}) \prod_{u \in \mathcal{U}} \Psi_{\text{C}}(\mathbf{y}, u) \tag{6.7}$$

and the potentials for the unigram factor are given by

$$\Psi_{\text{C}}(\mathbf{y}, u) = \exp\left\{ \sum_k \lambda_{\text{C}k} f_{\text{C}k}(\mathbf{y}, u) \right\} \tag{6.8}$$

whereas $\theta_{\text{C}} = \{\lambda_{\text{C}k}\}$ is the set of additional parameters for the classifier template. The feature function factorize into an indicator function $p_{\text{C}k}$ and an output function $q_{\text{C}k}$:

$$f_{\text{C}k}(\mathbf{y}, u) = p_{\text{C}k}(y_u) \cdot q_{\text{C}k}(\mathbf{y}, u) \tag{6.9}$$

Six different output functions are introduced:

$$\begin{aligned} q_{\text{e-consistent}}(\mathbf{y}, u) &= \begin{cases} 1 & \text{iff } y_u \neq y_{u+1} \wedge u \in \mathcal{R}_e(y_u) \\ 0 & \text{else} \end{cases} \\ q_{\text{e-project}}(\mathbf{y}, u) &= \begin{cases} 1 & \text{iff } y_u \neq \tilde{y} \wedge u \in \mathcal{R}_e(\tilde{y}) \\ 0 & \text{else} \end{cases} \\ q_{\text{e-suppress}}(\mathbf{y}, u) &= \begin{cases} 1 & \text{iff } y_u \neq y_{u+1} \wedge u \notin \mathcal{R}_e(y_u) \\ 0 & \text{else} \end{cases} \end{aligned} \tag{6.10}$$

[53] The different usage of $\mathbf{y}$ for the predicted sequence and the label configuration of the parameters deduces from the context.

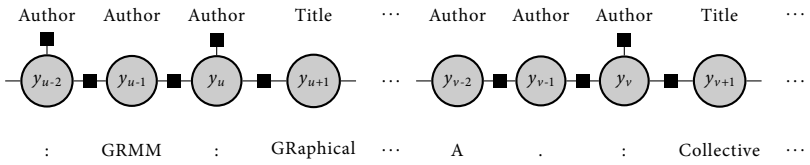


Figure 6.18: An excerpt of a comb-chain graph with erroneous labeling whereas only additional factors for the end of the author are displayed. The output functions indicate a missing end at position y_{u-2}, a surplus end at y_u and a consistent end at y_v

The output functions $q_{\text{b-consistent}}$, $q_{\text{b-project}}$ and $q_{\text{b-suppress}}$ are defined equivalently for the beginning of an entity. They reflect the result of the classification combined with the intermediate labeling: $q_{\text{e-consistent}}$ indicates a true positive, $q_{\text{e-project}}$ a false positive and $q_{\text{e-suppress}}$ a false negative classification compared to the label sequence. Together, these feature functions supply evidence, which parts of the label sequence agree with the consistency and which parts should be altered in order to gain a higher likelihood. The resulting graph of the model contains no loops and provides therefore fewer challenges for an inference mechanism.

The idea of comb-chain CRFs is summarized with an example for the segmentation of references (cf. Figure 6.18). Let a reference section be the input sequence. When unrolling the graph, the external model provides an intermediate labeling specifying the entities. A classifier is trained to detect the boundaries of the entities. The descriptive result of the classifier for the end of the author is, for example, a pattern like "A period followed by a colon". Now, the additional potentials with the output functions influence the model to assign a high likelihood to label sequences that confirm with the description of the classifier.

6.3.2 Skyp-chain CRFs

Skyp-chain CRFs are a variant of skip-chain CRFs (cf. Section 2.3.3.1). But instead of creating additional edges between labels, whose tokens are similar or identical, this approach adds long-range dependencies based on the patterns occurring in the predicted label sequence $\mathbf{y}$ and the classification result. Thus, the small modification of the name. Earlier work on skyp-edges without a distinct model for consistencies can be found in Toepfer et al. [195]. However, the implementation of the predecessor is flawed, which led to false results in its evaluation.

When applying skyp-chain CRFs for exploiting context-specific consistencies, two additional differences to published approaches for skip-chain CRFs or similar collective information extraction models can be identified:

1. There is no need to transfer local evidence to distant labels since this work already assumes a homogeneous composition of the entities.
2. Useful observation functions for the skip edges cannot be specified, because the relevance of certain properties is unknown.

First, the set of additional edges is defined that specify the positions of the long-range dependencies using the positions $\mathcal{U}_b$ and $\mathcal{U}_e$ of Equation 6.6.

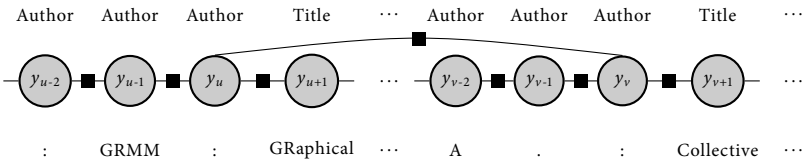


Figure 6.19: An excerpt of a skyp-chain graph with erroneous labeling. Only one additional edge for the end of the author is displayed. The likelihood of the sequence is decreased because only position $u-2$ and v but not u were identified as a boundary by the classifier.

$$\begin{aligned}\mathcal{E}_b &= \{(u,v) : u \neq v \wedge y_u = y_v \wedge u \in \mathcal{U}_b \wedge v \in \mathcal{U}_b\} \\ \mathcal{E}_e &= \{(u,v) : u \neq v \wedge y_u = y_v \wedge u \in \mathcal{U}_e \wedge v \in \mathcal{U}_e\} \\ \mathcal{E} &= \mathcal{E}_b \cup \mathcal{E}_e\end{aligned} \tag{6.11}$$

The set $\mathcal{E}_b$ contains edges that connect the start label of an entity with all other start labels of entities with the same type. The set $\mathcal{E}_e$ refers accordingly to the links between the end labels of entities. Further, this work introduces a parameter m_e for controlling the model complexity that restricts the maximal amount of additional long-range dependencies for each variable. E.g., for $m_e = 2$, a label is only connected to the closest previous and following boundary of the same entity type.

The skyp-chain approach extends the linear-chain model with additional potentials for edges defined in Equation 6.11. The conditional probability for the assignment of the label sequence is given by

$$p_\theta(\mathbf{y}|\mathbf{x}) = \frac{1}{Z(\mathbf{x})} \prod_{t=1}^{T} \Phi_{\mathrm{L}}(y_t, y_{t-1}, \mathbf{x}) \prod_{(u,v)\in\mathcal{E}} \Psi_{\mathrm{Y}}(\mathbf{y}, u, v)\,. \tag{6.12}$$

An example of an unrolled graph of this model is depicted in Figure 6.19. Similar to Equation 2.11, the additional potentials factorize to

$$\Psi_{\mathrm{Y}}(\mathbf{y}, u, v) = \exp\left\{\sum_k \lambda_{\mathrm{Y}k} f_{\mathrm{Y}k}(\mathbf{y}, u, v)\right\}, \tag{6.13}$$

resulting in the complete parameter set $\theta = \theta_{\mathrm{L}} \cup \theta_{\mathrm{Y}}$ with $\theta = \theta_{\mathrm{Y}} = \{\lambda_{\mathrm{Y}k}\}$ to be estimated for this model. In contrast to the skip-chain model, the feature functions depend on the complete (predicted) label sequence $\mathbf{y}$. The feature functions consist again of an indicator function for the label configuration, but not of an observation function on the input sequence. Instead, this work applies the output functions of Equation 6.10 separately for the source and destination of the skip edge.

The skyp-chain model is illustrated in an example for reference segmentation (cf. Figure 6.19). Let the input sequence be a reference section. When the graph of the model is unrolled during inference, a label assignment are calculated. During this process, long-range dependencies are considered, e.g., for the end of the author entities (cf. labels y_u and y_v in Figure 6.19). Due to

the additional potentials, label sequences with boundaries that are identified by the classifier as consistently structured become more likely. In Figure 6.19, the likelihood of the sequence is decreased in comparison to a graph with an additional edge between the labels y_{u-2} and y_v.

6.3.3 Parameter Estimation and Inference

The conditional probability $p_\theta(\mathbf{y}|\mathbf{x})$ is computed to decide which label sequence $\mathbf{y}$ is most likely for the observed token sequence $\mathbf{x}$, and to estimate the parameters θ of the model. The applied inference technique, tree based reparameterization (TRP) [200], is related to belief propagation and computes approximate marginals for loopy graphs. TRP is also used in [189] for the original skip-chain models. Unfortunately, severe convergence problems could be observed when applied on complex graph structures. The parameters θ of the models are obtained using training data $D = \left\{\mathbf{x}^{(i)}, \mathbf{y}^{(i)}\right\}_{i=1}^{N}$ and maximum a-posteriori estimation. The log likelihood $\mathcal{L}(\theta|D)$ of the model parameters given the training examples is optimized with the quasi-Newton method L-BFGS and a Gaussian prior on the parameters as in [189].

6.3.4 Experimental Results

This work demonstrates the advantages of the presented approach in a five-fold cross evaluation in two different real-world applications: The segmentation of references and the template extraction in curricula vitae (cf. Chapter 3). First, both domains and the real-world datasets are shortly recapitulated for convenience and then the settings of the evaluation are specified. Finally, the empirical results are presented and discussed.

6.3.4.1 Datasets

Two datasets are utilized in the evaluation of this work. The dataset *References* origins in a domain that is very popular for the evaluation of novel information extraction techniques (cf. [15, 160, 163, 183]), whereas the dataset *Curricula Vitae* belongs to classical information extraction problems of template extraction.

This dataset for the segmentation of references was introduced in [128] (cf. Section 6.2) and consists only of complete reference sections of real publications, mainly from the computer science domain. For the evaluation in this section, this work reduced the label set for the identification of the entities Author, Date, Title and Venue. The same features as in [128] (cf. Section 6.2) have been extracted.

The information extraction task in curricula vitae consists in identifying the time span and company for which the author of these documents worked in a stage of his or her life (employments). This information can be used to improve the search for suitable future employees for certain projects. The dataset consists of 68 German curricula vitae and is annotated with 896 companies or sectors[54] and 937 time spans in overall 921 stages of life. This work uses the label Date for the time span and the label Client for the companies or sectors. The feature set extends the feature set of the dataset *References* with additional domain-specific features like the number of the line, the position within a line and keywords for company prefixes/suffixes

[54] The authors of the curricula vitae sometimes anonymize the actual name of a company and replace it with the sector in which the company is located.

and date indicators. For the dataset *References* the F_1 score is presented combined for all labels, whereas the labels DATE and CLIENT are distinguished for *Curricula Vitae*.

6.3.4.2 Settings

All models are trained with identical settings. In order to minimize the model complexity of the skyp-chain approach, this work set $m_e = 2$. Eleven (for *References*) and twelve (for *Curricula Vitae*) manually selected features are used in a window of five tokens as attributes for the rule learner. The learnt descriptions had a maximum of three attributes and a minimum quality score of 0.01. For the dataset *References*, only the boundaries for the labels AUTHOR, DATE, TITLE are considered. The quality function F_1^{exp2} is utilized. The implementation of the CRFs is based on the GRMM package [188].

6.3.4.3 Results

The proposed models are compared to a linear-chain CRF (base line). This work has considered different variants of skip-chain CRFs, but none of them returned noteworthy results. As a consequence, the presented models are only compared to the base line.

The results of the five-fold cross evaluation are depicted in Table 6.5 for the dataset *References* and in Table 6.6 for the dataset *Curricula Vitae*. The comb-chain models achieve overall an average error reduction of over 30% and increase the measured averaged F_1 score by at least 1%, 9% for the label CLIENT. The skyp-chain model provides more challenges for the inference technique and is only able surpass the comb-chain results for the label DATE of the dataset *Curricula Vitae*. In the evaluation of the remaining label, the average error reduction is 14%.

If the comb-chain model is compared to the skyp-chain model, then it becomes apparent that the skyp-chain model with the applied inferencing technique TRP has no advantages when exploiting consistencies even at the cost of a computationally more expensive inference. Table 6.7 and Table 6.8 contain the average evaluation time for one fold. In general, it takes longer to train models with the larger dataset *Curricula Vitae*. The discrepancies of the results in the domain of references compared to the results of the stacked approach are explained in Section 6.4.

The evaluated results of the presented models have a valuable influence on real-world applications. An error reduction of 30% considerably improves the quality of automatically extracted entities and reduces the workload to correct possible errors. The reported increase of the accuracy and the corresponding error reduction of the presented models compete well with published approaches for collective information extraction, joint inference in information extraction or other models that exploit long-range dependencies.

6.4 Discussion

The experimental studies of this section show that context-specific consistencies can be exploited by Conditional Random Fields. The presented approaches are generic and can be applied for any appropriate domain with only minimal configuration decisions. The overall findings can be summarized with the expectation that the usage of context-specific consistencies is able to produce an average error reduction of 30% concerning the F_1 score. This improvement

References	All
Linear Chain	0.966
Comb Chain	0.976
Skyp Chain	0.972

Table 6.5: F_1 scores for the segmentation of references.

Curricula Vitae	Date	Client
Linear Chain	0.944	0.725
Comb Chain	0.962	0.814
Skyp Chain	0.962	0.764

Table 6.6: F_1 scores for template extraction in curricula vitae.

References	
Linear Chain	0.03h
Comb Chain	0.17h
Skyp Chain	0.53h

Table 6.7: Average time for one fold (*References*).

Curricula Vitae	
Linear Chain	0.11h
Comb Chain	0.27h
Skyp Chain	0.97h

Table 6.8: Average time for one fold (*Curricula Vitae*).

competes well with many approaches that exploit different asumptions and joint tasks and even outperforms many results of collective information extraction techniques.

The first experimental study investigated the capabilities of subgroup discovery for learning descriptions for context-specific consistencies in a document. The rules are able to identify a great amount of correct boundaries independently of the quality of the input. The evaluation showed also the limitation of the approach. If no valuable prediction is available, then no consistencies at all can be learnt. Another problem can be identified in documents where the set of features and properties are not sufficient for describing the aspects of consistencies. Both situations can cause a decrease of accuracy when the learnt consistencies are exploited in further models. The experimental evaluation is supplemented by a more realistic scenario where a Conditional Random Field with optimal configuration is applied for predicting the entities. The subgroup discovery task is still able to improve the identification of the boundaries.

The approaches based on Conditional Random Fields can be compared using different perspectives like the necessary effort for the implementation and the quality of their extractions. The stacked approach is clearly the most straightforward to implement since it only requires to connect two models with an intermediate phase for inducting the feature functions. The comb-chain and skyp-chain approach require a bit more effort to implement. They defined a new clique template that integrates the induction of the consistency, which requires a prediction of the entities. Thus, the utilized implementation of the Conditional Random Field needs to be adapted in order to either access the current model or for asking an external model for incrementally unrolling the graph during inference. When comparing the results of the different approaches, the skyp-chain model clearly falls back behind the other models due to the inference problems in its cyclic graph structure. Using more sophistical inference techniques like SampleRank [174], the model should be able to exploit its full potential. Differences between the stacked model and the comb-chain model in their current configuration consist mostly in the representation and integration of the information about the consistencies. Overall, the comb-chain model provides

some more advantages since it allows one potentially to incrementally unroll its graph structure dependent on its own prediction.

An interesting observation can be made when the results of the evaluation of the consistency learning task for curricula vitae is compared to the results of the comb-chain approach exploiting the consistencies in the same domain. The subgroups are able to achieve an error reduction of about 16% for detection the correct boundaries of clients and dates. The comb-chain approach is, however, able to achieve an average error reduction of about 30%. This improvement highlights a very nice feature of the presented approach. The description of the context-specific consistencies does not need to be perfect in order to achieve good results when exploiting them. If the subgroups classify to many tokens as boundaries of an entity, then the Conditional Random Field is able to take advantage of the correct ones. Additionally, in situations where the subgroups missed correct boundaries, the Conditional Random Field can still rely on its own mechanisms.

A closer look at the results of the stacked and the comb-chain approach for segmentation of references reveals that the results of the baseline and also the achieved scores of the presented techniques differ considerably. This discrepancy is caused by various factors. The experimental setting of the comb-chain approach defined a smaller set of entities. The information extraction task is limited to the identification of the author, title, date and venue. Thus, the learning task is much simpler and higher scores can be achieved. However, this variation of the setting is not the only reason. The comb-chain approach utilized another implementation of Conditional Random Fields and also a different inference technique. Furthermore, the models of the stacked approach are applied on the references separately and use transition-based descriptions for modeling the consistencies. In contrast to this, the higher-order models process the complete reference sections at once and utilize the boundaries of entities for describing the consistencies.

Chapter 7

Conclusion

This chapter provides a summary of the presented work and discusses its contribution. Furthermore, an outlook highlights different options for future work that will further improve the impact of exploiting context-specific consistencies in information extraction.

7.1 Summary

Information extraction addresses the identification of well-defined entities and relations in unstructured data and especially in textual documents. A vast amount of information is stored and exchanged in an unstructured representation since it is mainly intended to be interpreted by humans. In order to access the concealed information for analytic processes, it has to be transformed into a structured representation. Hence, information extraction has become a key component in the integration of textual data and can be considered as an umbrella term for many interesting tasks such as named entity recognition, sentiment analysis or knowledge extraction. Approaches to information extraction can roughly be divided into two main categories: approaches based on handcrafted rules and approaches based on statistical models trained in a supervised fashion.

When extracting entities and relations it is often assumed that the information in textual documents is independent and identically distributed (iid). However, many documents violate these assumptions. Especially semi-structured documents often contain a special form of long-range dependencies between entities. The context in which the textual data is created or written introduces a homogeneous composition of the entities. These dependencies are called context-specific consistencies in this work. If these consistencies are not taken into account, then the information extraction system faces a heterogeneous and inconsistent composition of the entities in the complete dataset. However, by considering the consistencies between entities within a context and processing those entities collectively, many labeling errors can be prevented.

This work provided an investigation of context-specific consistencies and presented different approaches for exploiting these consistencies in order to improve information extraction applications: approaches based on handcrafted rules and approaches based on probabilistic models using supervised machine learning. Furthermore, the rule-based system UIMA Ruta has been developed in order to support and improve the knowledge engineering process when writing rules sensitive to the consistencies. These contributions are shortly summarized in the following.

UIMA Ruta is a rule-based system for information extraction and for general natural language processing tasks. A special focus of the system lies on a compact rule language with a high expressiveness combined with strong development support. These attributes facilitate rapid development of rule-based applications and thus reduce one major bottleneck when handcrafting extraction knowledge: the time and costs of the engineering task. UIMA Ruta provides most of the features of related systems concerning language and tooling support. Furthermore, it introduces several new and useful elements that are not found in other systems. These include amongst others a coverage-based concept of visibility, powerful vertical matching or estimation of the rules' quality on unlabeled documents. UIMA Ruta has not been specifically created for exploiting context-specific consistencies. It is a useful general-purpose tool for many diverse use cases. In contrast to other systems, however, it provides some features that facilitate the integration of context-specific consistencies in rule-based applications. Amongst others, these include language elements rather unknown to rule languages. Lists and variables help to model consistencies with rule sets and allow one to integrate dynamic knowledge about the currently processed document. An example for this fact is the usage of variables in the matching condition of a rule in order to process an entity dependent on the dominant composition of all entities. Another important aspect consists in the availability of different engineering approaches. Due to the high expressiveness of its language, UIMA Ruta supports always more than one approach to solve an annotation problem, which can be essential for dealing with different aspects of consistencies. Besides these properties, efficient and effective engineering is also helpful when handcrafting rule sets for context-specific consistencies and leads to an improved engineering experience in general.

Three case studies highlight different engineering approaches that enable a knowledge engineer to integrate the consistencies in rule-based applications. In the first case study, simple precision-driven rules are extended with rules for finding additional entities that share a composition similar to detected ones. This approach is very effective and efficient since the utilized rules are easy and fast to engineer. Even if the resulting applications are not sufficient for solving real-world information extraction tasks, they provide many advantages. The approach can be utilized for creating prototypes, which perform almost as good as corresponding applications, or for a fast analysis of a domain, e.g., by extracting lists of entities. The results can then be applied to accelerate the development of the actual application. The engineering approach is evaluated for the identification of companies in curricula vitae and for detecting headlines in clinical discharge letters. Both rule sets are able to achieve an F_1 score of over 0.97 for unseen documents in the respective domain and have been created in less than two hours. The experimental setup is limited but reflects realistic conditions in real-world scenarios. A feasibility study should be performed with a limited amount of documents, which prevents the usage of supervised machine learning approaches. The second case study considers a different domain and a more sophisticated approach for exploiting context-specific consistencies. Here, rules are applied in a transformation-based manner in order to segment scientific references. An initial set of rules extracts interesting entities like author, title and date. Then, rules investigate the composition of the entities and create a model of different aspects of the occurring context-specific consistencies. This model is utilized by transformation-based rules in order to modify the initial entities. The rules reclassify and change the offsets of the entities until they confirm with the dominant composition of entities of the same type in the current document. The procedure greatly increases the accuracy of the application if the assumption about the consistent composition of entities is

fulfilled. This case study illustrates some advantages of the rule language of UIMA Ruta. The model of consistencies is specified using only language elements of UIMA Ruta. The properties of the consistent composition are stored in variables and thus can be directly integrated in rules. Furthermore, the necessary transformation can be efficiently specified in UIMA Ruta since the language allows one to modify arbitrary annotations with specialized actions. The combination of the three phases is able to achieve an F_1 score of 0.997 for consistently formatted references. While the experimental setup is limited, it highlights the power of this engineering approach. The last case study investigates the usage of consistencies in a complete application. The rules of this case study identify and categorize sections in clinical discharge letters. In these sections, specialized information extraction models are applied in order to populate a clinical data warehouse. The rules utilize different approaches including semantic keywords, formatting and also context-specific consistencies in order to remove false positive headlines in a set of potential candidates. Thus, the consistencies are applied in order to increase the precision. Additional rules utilize the headlines in order to identify the sections and their categories. The rules are able to achieve an F_1 score of 0.992 in a test set of 200 unseen documents. Overall, this case study highlights the usefulness of exploiting context-specific consistencies in a throughout engineered application. Although the rules have been optimized to perform in a best possible manner, the integration of the consistencies was still able to further improve the accuracy.

The part of this work that considers machine learning starts with techniques how to learn a model of the context-specific consistencies. The general approach is based on in utilizing binary classifiers in order to describe specific aspects of the consistencies like the boundaries of entities or transitions between specific types of entities. This approach provides several advantages compared to related work. Learning a model represented by a set of classifiers enables the usage of a combination of features instead of only one specific property. Thus, the consistencies can be described more precisely. Another advantage of a distinct model results from the fact that this model can be applied not only to make statements about how consistent an entity is, but also what other, not yet considered positions fulfill the different aspects of consistency. The modeling of the context-specific consistencies is evaluated separately from the actual information extraction task in two experimental settings. In both settings, the domains of curricula vitae and references are considered. The first experiment investigates the ability of subgroup discovery to classify the correct boundaries of entities given erroneous training data. For this purpose, a gold standard dataset is incrementally deteriorated by replacing a correct boundary by an incorrect one. The learnt rules are able to greatly improve the F_1 score and even classify the boundaries of the entities correctly despite of only minimal amount of correct input data. The second experiment investigates the performance of subgroup discovery on more realistic predictions. The learnt rules achieve an error reduction of 16%-30% for the classification of boundaries compared to the prediction provided by a five-fold Conditional Random Field.

Three approaches based on Conditional Random Fields are presented that utilize the learnt model in order to exploit the context-specific consistencies. The first approach applies stacked Conditional Random Fields with two phases. The initial model provides a prediction of the entities in a document, which is utilized in order to learn a model of the consistencies. The stacked model is extended with additional feature functions that provide static semantics for the model, but change their manifestation dependent on the currently induced consistencies. This is achieved by directly utilizing the rules of the consistency model as feature functions. The approach is evaluated for the segmentation of references and is able to achieve an F_1

score of 0.940 in a cross fold setting. This constitutes an error reduction of 30% compared to a baseline Conditional Random Field. The remaining approaches based on Conditional Random Fields add additional factors dependent on the learnt consistency model. Both models utilize a prediction of entities provided by an external Conditional Random Field. Thus, their approach is similar to stacking the models, but they also support an incremental unfolding of the graph during inference. The first model called comb-chain extends the probabilistic graph with unigram factors on the positions indicated by the learnt rules. New feature functions represent consistent positions (true positive) as well as missing (false positive) and additional (false negative) positions compared to the prediction. The second model is called skyp-chain and extends the graph structures with long-range dependencies between interesting positions. The same set of feature functions is applied as before. The experimental results for the segmentation of references and entity extraction in curricula vitae again indicate an error reduction of up to 30% compared to a baseline Conditional Random Field. The comb-chain model overall performs better due to inference problems in the more complex graph structure of the skyp-chain model. It achieves an F_1 score of 0.976 (references), 0.962 (dates in curricula vitae) and 0.814 (companies in curricula vitae). The discrepancy of the results concerning the segmentation of references compared to the results of the stacked approach is mainly caused by the usage of a different set of labels and specification of an instance.

In summary, the presented approaches provide practical solutions for exploiting context-specific consistencies that have considerable impact in real-world applications. UIMA Ruta enables the knowledge engineer to integrate the consistencies in rapid prototypes as well as in complex rule-based applications. The machine learning approaches are able to achieve an error reduction of 30% compared to a Conditional Random Field while still relying on common techniques. Thus, the presented models compete well with published approaches for improved models, e.g., collective information extraction, joint inference in information extraction or other models that exploit long-range dependencies. A direct comparison of approaches based on handcrafted rules and approaches based on machine learning is problematic, because the quality of the rules is inherently influenced by the human factor. The experimental evaluations are called case studies because they are hardly reproducible leading to same results. Either a different knowledge engineer with other training or capabilities is employed, or the initial knowledge engineer gained knowledge just by the work she performed. This fact prevents an evaluation in a cross-fold setting. The machine learning approaches are easier to evaluate. Given the labeled data, the extracted features and the algorithms, the same results can be reproduced. The results of the rule-based case studies seem to be superior to the results of the machine learning evaluations if only the absolute scores are considered. Nevertheless, a conclusion that rule-based approaches perform better would be wrong. The rule engineer is able to integrate extraction knowledge beyond the given examples. She can think of possible situations that might occur in the other documents like the test set, and specify the rules accordingly. The machine learning approaches solely rely on the labeled examples. Both approaches have been evaluated for the segmentation of references and entity extraction in curricula vitae. However, the documents applied for evaluating the machine learning approaches are much more realistic and problematic. While the examples used in the rule-based case studies are well-formatted, the labeled entities in the machine learning evaluation not always share a consistent composition. Either the author of the document accidently changed the layout, did not care about the structure, or preprocessing steps broke the consistencies. Nevertheless, the approaches performed well in

this realistic setting. The rule-based case studies also contain a domain with documents where the context-specific consistencies are not always fulfilled: the clinical discharge letters. However, the results in this domain indicate an F_1 score of 0.992, a result that highlights that potential problems can be solved with rule-based approaches.

7.2 Outlook

The presented approaches in this work already provide well-suited techniques for exploiting context-specific consistencies in information extraction. However, there remain many possibilities to further improve these approaches or to transfer specific ideas to other approaches, tasks and methodologies. Besides these improvements, the general idea and techniques can also be applied in different domains.

UIMA Ruta provides language elements and functionality in order to engineer rules sensitive to context-specific consistencies in an efficient and effective manner. The engineering process can, however, be further improved by providing more complex actions like operations on lists. Currently, these include only the selection of an element in a list that occurs most often, which has been applied in order to determine the dominant properties of consistency. It will help the knowledge engineer to rapidly specify dependencies between entities without relying on additional functionality defined in another language like Java. Besides improvements concerning context-specific consistencies, many interesting options for future work can be identified. These include enhancements of the runtime performance by compiling sets of rules into finite-state transducers or by using optimized execution plans that extend dynamic anchoring. Another interesting direction is the further development of rule learning algorithms that induce human-readable rule sets. The KEP algorithms introduced in Section 4.3.6 takes advantage of rule engineering patterns and provides a first step in this direction.

The approaches of this work illustrated their advantages separately from each other. Each approach provides distinct advantages and is suited for different settings. A combination of the approaches has the potential to benefit from the different strengths and to further increase the effect of exploiting context-specific consistencies. Examples for possible combinations are manifold. The knowledge engineering approaches relied solely on handcrafted extraction knowledge, which also includes the induction of the consistencies occurring in the currently processed document. Thus, the model of consistencies is rather limited compared to the presented approach based on binary classifiers. This more powerful technique can also be utilized in rule-based applications. After acquiring an initial prediction of entities, the consistencies are modeled with machine learning techniques (subgroup discovery) instead of using handcrafted rules. Then, this improved description can be utilized by other rules in order to identify additional entities, filter inconsistent entities, or repair entities in order to confirm with the learnt consistencies. Especially the improved quality function will provide many advantages for approaches based on a small amount of highly confident entities. Another example is the combination of approaches based on Conditional Random Fields with transformation-based rules. These rules can simply be applied after a model like the comb-chain approach processed the document. The transformation-based rules operate on the entities extracted by the probabilistic model. The rules face, for example, fewer problems to express long-range dependencies and thus are able to remove a variety of the remaining errors.

This work applied subgroup discovery in order to learn a model of the context-specific consistencies. This technique works well, but there are always options to improve this task in general or for a specific domain. Statistical models are able to include a large amount of features in their classification decision and potentially provide the capability to model the consistencies more precisely. An interesting option is the integration of background knowledge, e.g., the expected amount of entities, in the optimization objective when training the statistical model. Furthermore, additional documents including labeled and unlabeled examples can be utilized amongst others for selecting or pruning the set of applied features.

Concerning the approaches based on Conditional Random Fields, more complex models with sophisticated inference techniques have the potential to perform better than the presented models. The work of Gupta et al. [94] about the properties-based collective inference and the work of Blei et al. [25] on generative models with scope present interesting extensions. They either provide inference techniques better suited for complex graph structures or integrate the induction of the consistencies directly in the graphical model. However, ideas of this work should be integrated in these models in order to overcome their drawbacks, which are described in Section 3.4.1.

The knowledge engineering approach based on transformations presents a potent methodology for exploiting context-specific consistencies in information extraction. Here, the stacked rules do not extract the entities anew, but only correct potential errors of the given entities. This idea of applying transformations can also be utilized in approaches based on machine learning. A first step consists in automatically inducing the transformations in a supervised fashion with the TraBaL algorithm presented in Section 4.3.6. This rule induction algorithm is able to learn transformation-based rules for a specific component given labeled examples. By providing the information of the model for context-specific consistencies, the induced rules are able to take advantage of the identified consistent and inconsistent positions. Even more promising is the development of a statistical model that is able to perform transformations instead of sequence labeling. It could consist in a set of Support Vector Machines that specify the transformations or a graphical model with a graph structure that does not reflect the token sequence, but chunks of label shifts. This approach potentially provides the advantages of including a large amount of features and does not need to repeat the work of an initial model.

Context-specific consistencies can also be exploited for other tasks. One idea is the estimation of a component's quality by analyzing how consistent the extracted entities are. The approach for learning a model of consistencies can be utilized in the constraint-driven evaluation framework presented in Section 4.3.5. Here, a constraint rates, for example, the violations compared to the learnt consistencies. The basic idea has already been used in the creation and validation of gold standard documents. Under the assumption that entities share a similar composition, the model of context specific consistencies is able to point out false positive and false negative entities. Several erroneous entities created by human annotators have been found using this approach in the labeled datasets for references and curricula vitae.

When pursuing this idea further, one ends up in improving the parameter estimation of probabilistic models. Several approaches for semi-supervised training of Conditional Random Fields have been proposed (cf. [16, 15, 146]). The approaches apply background knowledge and expectations in order to estimate the parameters using unlabeled documents. These techniques can be extended by constraints that are sensitive to context-specific consistencies. In suitable domains where the assumptions about the consistencies are fulfilled, the estimation of the model

parameters can be supported by measuring how consistent the predicted label sequence is.

All approaches and domains presented in this work assume that there exists a trivial mapping between instances and their context. In most domains, the context is given by the document and all instances or entities are automatically associated with this context. It is possible that this assignment is not straightforwardly given in some domains. Thus, approaches need to be developed, which are able to associate a set of instances to a given context or define the contexts from scratch. This task resembles clustering or classification of documents whereas one cluster or category refers to one context.

Bibliography

[1] Muhammad Tanvir Afzal, Hermann Maurer, Wolf-Tilo Balke, and Narayanan Kulathuramaiyer. Rule based Autonomous Citation Mining with TIERL. *Journal of Digital Information Management*, 8(3):196–204, 2010.

[2] Rakesh Agrawal, Ramakrishnan Srikant, et al. Fast Algorithms for Mining Association Rules. In *Proc. 20th Int. Conf. Very Large Data Bases, VLDB*, volume 1215, pages 487–499, 1994.

[3] Yuan An, Jeannette Janssen, and Evangelos E Milios. Characterizing and Mining the Citation Graph of the Computer Science Literature. *Knowledge and Information Systems*, 6(6):664–678, 2004.

[4] S. Anzaroot and A. McCallum. A New Dataset for Fine-Grained Citation Field Extraction. In *ICML Workshop on Peer Reviewing and Publishing Models (PEER)*, 2013.

[5] C. Aone, L. Halverson, T. Hampton, and M. Ramos-Santacruz. SRA: Description of the IE2 System Used for MUC-7. In *Seventh Message Understanding Conference (MUC-7): Proceedings of a Conference Held in Fairfax, Virginia, April 29 - May 1, 1998*, pages 1–14, 1998.

[6] Douglas E. Appelt and Boyan Onyshkevych. The Common Pattern Specification Language. In *Proceedings of a Workshop on Held at Baltimore, Maryland: October 13-15, 1998*, TIPSTER '98, pages 23–30, Stroudsburg, PA, USA, 1998. Association for Computational Linguistics.

[7] Andrew Arnold and William W Cohen. Intra-document Structural Frequency Features for Semi-supervised Domain Adaptation. In *Proceedings of the 17th ACM Conference on Information and knowledge management*, pages 1291–1300. ACM, 2008.

[8] Martin Atzmueller, Joachim Baumeister, Peter Kluegl, and Frank Puppe. Rapid Knowledge Capture Using Subgroup Discovery with Incremental Refinement. In *K-CAP '07: Proceedings of the 4th International Conference on Knowledge Capture*, pages 31–38, 2007.

[9] Martin Atzmueller, Peter Kluegl, and Frank Puppe. Rule-Based Information Extraction for Structured Data Acquisition using TextMarker. In Joachim Baumeister and Martin Atzmueller, editors, *LWA-2008 (Special Track on Knowledge Discovery and Machine Learning)*, pages 1–7, 2008.

[10] Hannah Bast and Claudius Korzen. The Icecite Research Paper Management System. In *Web Information Systems Engineering–WISE 2013*, pages 396–409. Springer, 2013.

[11] Iyad Batal and Milos Hauskrecht. Constructing Classification Features using Minimal Predictive Patterns. In *Proceedings of the 19th ACM Int. Conf. on Information and Knowledge Management*, CIKM '10, pages 869–878. ACM, 2010.

[12] Joachim Baumeister, Martin Atzmueller, Peter Kluegl, and Frank Puppe. Conservative and Creative Strategies for the Refinement of Scoring Rules. In *FLAIRS'06: Proceedings of the 19th International Florida Artificial Intelligence Research Society Conference*, pages 408–413. AAAI Press, 2006.

[13] Philip-Daniel Beck. Identifikation und Klassifikation von Abschnitten in Arztbriefen (in German). Master's thesis, University of Würzburg, 2013.

[14] Markus Becker, Witold Drozdzynski, Hans-Ulrich Krieger, Jakub Piskorski, Ulrich Schäfer, and Feiyu Xu. SProUT - Shallow Processing with Typed Feature Structures and Unification. In *Proceedings of the International Conference on NLP (ICON 2002). December 18-21*, Mumbai, India, 2002.

[15] Kedar Bellare, Gregory Druck, and Andrew McCallum. Alternating Projections for Learning with Expectation Constraints. In *Proceedings of the Twenty-Fifth Conference on Uncertainty in AI*, pages 43–50. AUAI Press, 2009.

[16] Kedar Bellare and Andrew McCallum. Generalized Expectation Criteria for Bootstrapping Extractors using Record-text Alignment. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 1*, pages 131–140. Association for Computational Linguistics, 2009.

[17] Dimitri P Bertsekas. *Nonlinear programming*. Athena Scientific, 1999.

[18] Indrajit Bhattacharya and Lise Getoor. Collective Entity Resolution in Relational Data. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 1(1):5, 2007.

[19] Mikhail Bilenko and Raymond J Mooney. Adaptive Duplicate Detection using Learnable String Similarity Measures. In *Proceedings of the ninth ACM SIGKDD international Conference on Knowledge discovery and data mining*, pages 39–48. ACM, 2003.

[20] Steven Bird, Robert Dale, Bonnie Dorr, Bryan Gibson, Mark Joseph, Min-Yen Kan, Dongwon Lee, Brett Powley, Dragomir Radev, and Yee Fan Tan. The ACL Anthology Reference Corpus: A Reference Dataset for Bibliographic Research in Computational Linguistics. In *Proceedings of the Sixth International Language Resources and Evaluation (LREC'08)*, Marrakech, Morocco, May 2008. European Language Resources Association (ELRA).

[21] Steven Bird, David Day, John Garofolo, John Henderson, Christophe Laprun, and Mark Liberman. ATLAS: A Flexible and Extensible Architecture for Linguistic Annotation, 2000. cite arxiv:cs/0007022.

[22] Steven Bird, Ewan Klein, and Edward Loper. *Natural Language Processing with Python*. O'Reilly Media, Inc., 1st edition, 2009.

[23] Christopher M Bishop. *Pattern Recognition and Machine Learning*, volume 1. Springer New York, 2006.

[24] William J. Black, John McNaught, Argyrios Vasilakopoulos, Kalliopi Zervanou, Babis Theodoulidis, and Fabio Rinaldi. CAFETIERE: Conceptual Annotations for Facts, Events, Terms, Individual Entities and RElations. Technical report, Department of Computation, UMIST, Jan 2005. Parmenides Technical Report, TR-U4.3.1.

[25] David M Blei, J Andrew Bagnell, and Andrew K McCallum. Learning with Scope, with Application to Information Extraction and Classification. In *Proceedings of the Eighteenth Conference on Uncertainty in artificial intelligence*, pages 53–60. Morgan Kaufmann Publishers Inc., 2002.

[26] Branimir Boguraev and Mary Neff. A Framework for Traversing Dense Annotation Lattices. *Language Resources and Evaluation*, 44(3):183–203, 2010.

[27] Branimir K. Boguraev and Mary S. Neff. An Annotation-Based Finite-State System for UIMA: Pattern Matching over Annotations. Technical report, IBM T.J. Watson Research Center, P.O. Box 704, Yorktown Heights, NY 10598, USA, December 2006.

[28] Philip Bohannon, Srujana Merugu, Cong Yu, Vipul Agarwal, Pedro DeRose, Arun Iyer, Ankur Jain, Vinay Kakade, Mridul Muralidharan, Raghu Ramakrishnan, and Warren Shen. Purple SOX Extraction Management System. *SIGMOD Rec.*, 37(4):21–27, March 2009.

[29] Uldis Bojārs and John G Breslin. ResumeRDF: Expressing Skill Information on the Semantic Web. In *1 st International ExpertFinder Workshop*, 2007.

[30] Levent Bolelli, Seyda Ertekin, and C Lee Giles. Clustering Scientific Literature using Sparse Citation Graph Analysis. In *Knowledge Discovery in Databases: PKDD 2006*, pages 30–41. Springer, 2006.

[31] Kalina Bontcheva, Valentin Tablan, Diana Maynard, and Hamish Cunningham. Evolving GATE to Meet New Challenges in Language Engineering. *Natural Language Engineering*, 10(3/4):349–373, 2004.

[32] Michael Borst. Knowledge Engineering Patterns for Information Extraction Rule Learning. Bachelor thesis, University of Würzburg, 2010.

[33] Leo Breiman. Random Forests. *Machine learning*, 45(1):5–32, 2001.

[34] Eric Brill. Transformation-Based Error-Driven Learning and Natural Language Processing: A Case Study in Part-of-Speech Tagging. *Computational Linguistics*, 21(4):543–565, 1995.

[35] Razvan Bunescu and Raymond J. Mooney. Collective Information Extraction with Relational Markov Networks. In *Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics*, ACL '04, Stroudsburg, PA, USA, 2004. Association for Computational Linguistics.

[36] Mary Elaine Califf and Raymond J. Mooney. Bottom-up Relational Learning of Pattern Matching Rules for Information Extraction. *Journal of Machine Learning Research*, 4:177–210, 2003.

[37] Ulrich Callmeier, Andreas Eisele, Ulrich Schäfer, and Melanie Siegel. The DeepThought Core Architecture Framework. In Maria Teresa Lino, Maria Francisca Xavier, Fatima Ferreira, and Raquel Silva, editors, *Proceedings of the 4th International Conference on Language Resources and Evaluation*, pages 1205–1208. ELRA, European Language Resources Association, 2004.

[38] Andrew Carlson, Justin Betteridge, Bryan Kisiel, Burr Settles, Estevam R Hruschka Jr, and Tom M Mitchell. Toward an Architecture for Never-Ending Language Learning. In *AAAI*, volume 5, page 3, 2010.

[39] George Casella and Edward I George. Explaining the Gibbs Sampler. *The American Statistician*, 46(3):167–174, 1992.

[40] Chien-Chih Chen, Kai-Hsiang Yang, Hung-Yu Kao, and Jan-Ming Ho. BibPro: A citation Parser based on Sequence Alignment Techniques. In *Advanced Information Networking and Applications-Workshops, 2008. AINAW 2008. 22nd International Conference on*, pages 1175–1180. IEEE, 2008.

[41] Fajun Chen. Learning Information Extraction Patterns. Master's thesis, Iowa State University, 2000.

[42] Laura Chiticariu, Vivian Chu, Sajib Dasgupta, Thilo W. Goetz, Howard Ho, Rajasekar Krishnamurthy, Alexander Lang, Yunyao Li, Bin Liu, Sriram Raghavan, Frederick R. Reiss, Shivakumar Vaithyanathan, and Huaiyu Zhu. The SystemT IDE: An Integrated Development Environment for Information Extraction Rules. In *Proceedings of the 2011 ACM SIGMOD International Conference on Management of Data*, SIGMOD '11, pages 1291–1294, New York, NY, USA, 2011. ACM.

[43] Laura Chiticariu, Rajasekar Krishnamurthy, Yunyao Li, Sriram Raghavan, Frederick R. Reiss, and Shivakumar Vaithyanathan. SystemT: An Algebraic Approach to Declarative Information Extraction. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, ACL '10, pages 128–137, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics.

[44] Laura Chiticariu, Yunyao Li, and Frederick R. Reiss. Rule-based Information Extraction is Dead! Long Live Rule-based Information Extraction Systems! In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 827–832, Seattle, Washington, USA, October 2013. Association for Computational Linguistics.

[45] F. Ciravegna. $(LP)^2$, Rule Induction for Information Extraction Using Linguistic Constraints. Technical Report CS-03-07, Department of Computer Science, University of Sheffield, Sheffield, 2003.

[46] Fabio Ciravegna and Alberto Lavelli. LearningPinocchio Adaptive Information Extraction for Real World Applications. In *Proceedings of 3rd Romand Workshop*, July 2001. Frascati, Italy.

[47] William W. Cohen. Fast Effective Rule Induction. In *Proceedings of the Twelfth Int. Conference on Machine Learning*, pages 115–123. Morgan Kaufmann, 1995.

[48] Ronald Cornet and Nicolette de Keizer. Forty Years of SNOMED: a Literature Review. *BMC medical informatics and decision making*, 8(Suppl 1):S2, 2008.

[49] Corinna Cortes and Vladimir Vapnik. Support-Vector Networks. *Machine Learning*, 20(3):273–297, 1995.

[50] Eli Cortez, Altigran S. da Silva, Marcos André Gonçalves, Filipe Mesquita, and Edleno S. de Moura. FLUX-CIM: Flexible Unsupervised Extraction of Citation Metadata. In *JCDL '07: Proceedings of the 7th ACM/IEEE-CS joint Conference on Digital libraries*, pages 215–224, New York, NY, USA, 2007. ACM.

[51] Roger A Cote and Stanley Robboy. Progress in Medical Information Management: Systematized Nomenclature of Medicine (SNOMED). *Jama*, 243(8):756–762, 1980.

[52] Isaac Councill, C Lee Giles, and Min-Yen Kan. ParsCit: an Open-source CRF Reference String Parsing Package. In *Proceedings of the Sixth International Language Resources and Evaluation (LREC'08)*, Marrakech, Morocco, 2008. ELRA.

[53] H. Cunningham, D. Maynard, K. Bontcheva, and V. Tablan. GATE: an Architecture for Development of Robust HLT Applications. In *Proceedings of the 40th Anniversary Meeting of the Association for Computational Linguistics (ACL)*, Philadelphia, USA, 2002.

[54] H. Cunningham, D. Maynard, and V. Tablan. JAPE: a Java Annotation Patterns Engine (Second Edition). Research Memorandum CS–00–10, Department of Computer Science, University of Sheffield, November 2000.

[55] Hamish Cunningham. Indexing and Querying Linguistic Metadata and Document Content. *Recent Advances in Natural Language Processing IV: Selected papers from RANLP 2005*, 292:35, 2007.

[56] Hamish Cunningham, Allan Hanbury, and Stefan Rüger. Scaling up High-Value Retrieval to Medium-Volume Data. In Hamish Cunningham, Allan Hanbury, and Stefan Rüger, editors, *Advances in Multidisciplinary Retrieval (the 1st Information Retrieval Facility Conference)*, Lecture Notes in Computer Science, Volume 6107, Vienna, Austria, May 2010. Springer.

[57] Hamish Cunningham, Diana Maynard, Kalina Bontcheva, and Valentin Tablan. GATE: A Framework and Graphical Development Environment for Robust NLP Tools and Applications. In *Proceedings of the 40th Anniversary Meeting of the Association for Computational Linguistics (ACL'02)*, 2002.

[58] Hamish Cunningham, Diana Maynard, Kalina Bontcheva, Valentin Tablan, Niraj Aswani, Ian Roberts, Genevieve Gorrell, Adam Funk, Angus Roberts, Danica Damljanovic, Thomas Heitz, Mark A. Greenwood, Horacio Saggion, Johann Petrak, Yaoyong Li, and Wim Peters. *Text Processing with GATE (Version 6)*. 2011.

[59] Hamish Cunningham, Valentin Tablan, Ian Roberts, Mark A. Greenwood, and Niraj Aswani. Information Extraction and Semantic Annotation for Multi-Paradigm Information Management. In Mihai Lupu, Katja Mayer, John Tait, and Anthony J. Trippe, editors, *Current Challenges in Patent Information Retrieval*, volume 29 of *The Information Retrieval Series*. Springer, 2011.

[60] Min-Yuh Day, Richard Tzong-Han Tsai, Cheng-Lung Sung, Chiu-Chen Hsieh, Cheng-Wei Lee, Shih-Hung Wu, Kun-Pin Wu, Chorng-Shyong Ong, and Wen-Lian Hsu. Reference Metadata Extraction using a Hierarchical Knowledge Representation Framework. *Decis. Support Syst.*, 43(1):152–167, 2007.

[61] Luciano Del Corro and Rainer Gemulla. ClausIE: Clause-based Open Information Extraction. In *Proceedings of the 22nd international Conference on World Wide Web*, pages 355–366. International World Wide Web Conferences Steering Committee, 2013.

[62] Joshua C. Denny, Anderson Spickard III, Kevin B. Johnson, Neeraja B. Peterson, Josh F. Peterson, and Randolph A. Miller. Evaluation of a Method to Identify and Categorize Section Headers in Clinical Documents. *JAMIA*, 16(6):806–815, 2009.

[63] Joshua C Denny, Randolph A Miller, Kevin B Johnson, and Anderson Spickard III. Development and Evaluation of a Clinical Note Section Header Terminology. In *AMIA Annual Symposium proceedings*, volume 2008, page 156. American Medical Informatics Association, 2008.

[64] Ying Ding. Applying Weighted PageRank to Author Citation Networks. *Journal of the American Society for Information Science and Technology*, 62(2):236–245, 2011.

[65] Ying Ding, Gobinda Chowdhury, and Schubert Foo. Template Mining for the Extraction of Citation from Digital Documents. In *Proceedings of the Second Asian Digital Library Conference, Taiwan*, pages 47–62, 1999.

[66] AnHai Doan, Luis Granavo, Raghu Ramakrishnan, and Shivakumar Vaithyanathan, editors. *Special Issue on Managing Information Extraction*, volume 37 of *SIGMOD Record*, 2008.

[67] Mike Dowman, Valentin Tablan, Hamish Cunningham, and Borislav Popov. Web-Assisted Annotation, Semantic Indexing and Search of Television and Radio News. In *Proceedings of the 14th International World Wide Web Conference*, Chiba, Japan, 2005.

[68] Witold Drozdzynski, Hans-Ulrich Krieger, Jakub Piskorski, Ulrich Schäfer, and Feiyu Xu. Shallow Processing with Unification and Typed Feature Structures - Foundations and Applications. *Künstliche Intelligenz*, 1(1):17–23, 2004.

[69] Benjamin Eckstein, Peter Kluegl, and Frank Puppe. Towards Learning Error-Driven Transformations for Information Extraction. In *Workshop Notes of the LWA 2011 - Learning, Knowledge, Adaptation*, 2011.

[70] Sean R Eddy. Hidden Markov Models. *Current opinion in structural biology*, 6(3):361–365, 1996.

[71] Duygu Çelik and Atilla Elçi. An Ontology-Based Information Extraction Approach for Résumés. In Qiaohong Zu, Bo Hu, and Atilla Elçi, editors, *Pervasive Computing and the Networked World*, volume 7719 of *Lecture Notes in Computer Science*, pages 165–179. Springer Berlin Heidelberg, 2013.

[72] Robert J Elliott, Lakhdar Aggoun, and John B Moore. *Hidden Markov Models*. Springer, 1994.

[73] Oren Etzioni, Michele Banko, Stephen Soderland, and Daniel S. Weld. Open Information Extraction from the Web. *Commun. ACM*, 51(12):68–74, December 2008.

[74] Anthony Fader, Stephen Soderland, and Oren Etzioni. Identifying Relations for Open Information Extraction. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1535–1545. Association for Computational Linguistics, 2011.

[75] Ronald Fagin, Benny Kimelfeld, Frederick Reiss, and Stijn Vansummeren. Spanners: A Formal Framework for Information Extraction. In *Proceedings of the 32Nd Symposium on Principles of Database Systems*, PODS '13, pages 37–48, New York, NY, USA, 2013. ACM.

[76] Ronen Feldman and James Sanger. *Text Mining Handbook: Advanced Approaches in Analyzing Unstructured Data*. Cambridge University Press, New York, NY, USA, 2006.

[77] David Ferrucci and Adam Lally. UIMA: An Architectural Approach to Unstructured Information Processing in the Corporate Research Environment. *Natural Language Engineering*, 10(3/4):327–348, 2004.

[78] David A. Ferrucci, Eric W. Brown, Jennifer Chu-Carroll, James Fan, David Gondek, Aditya Kalyanpur, Adam Lally, J. William Murdock, Eric Nyberg, John M. Prager, Nico Schlaefer, and Christopher A. Welty. Building Watson: An Overview of the DeepQA Project. *AI Magazine*, 31(3):59–79, 2010.

[79] Georg Fette, Maximilian Ertl, Anja Wörner, Peter Kluegl, Stefan Störk, and Frank Puppe. Information Extraction from Unstructured Electronic Health Records and Integration into a Data Warehouse. In *GI-Jahrestagung*, pages 1237–1251, 2012.

[80] Georg Fette, Peter Kluegl, Maximilian Ertl, Stefan Störk, and Frank Puppe. Information Extraction from Echocardiography Records. In *Workshop Notes of the LWA 2011 - Learning, Knowledge, Adaptation*, 2011.

[81] Jenny Rose Finkel, Trond Grenager, and Christopher Manning. Incorporating non-local Information into Information Extraction Systems by Gibbs Sampling. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, ACL '05, pages 363–370, Stroudsburg, PA, USA, 2005. Association for Computational Linguistics.

[82] Dayne Freitag. Machine Learning for Information Extraction in Informal Domains. *Machine Learning*, 39(2):169–202, 2000.

[83] Dayne Freitag and Nicholas Kushmerick. Boosted Wrapper Induction. In *AAAI/IAAI*, pages 577–583, 2000.

[84] Pablo Gamallo, Marcos Garcia, and Santiago Fernández-Lanza. Dependency-based Open Information Extraction. In *Proceedings of the Joint Workshop on Unsupervised and Semi-Supervised Learning in NLP*, pages 10–18. Association for Computational Linguistics, 2012.

[85] Liangcai Gao, Zhi Tang, and Xiaofan Lin. Cebbip: A Parser of Bibliographic Information in Chinese Electronic Books. In *Proceedings of the 9th ACM/IEEE-CS joint Conference on Digital libraries*, pages 73–76. ACM, 2009.

[86] C Lee Giles, Kurt D Bollacker, and Steve Lawrence. CiteSeer: An Automatic Citation Indexing System. In *Proceedings of the third ACM Conference on Digital libraries*, pages 89–98. ACM, 1998.

[87] Michael Granitzer, Maya Hristakeva, Robert Knight, Kris Jack, and Roman Kern. A Comparison of Layout based Bibliographic Metadata Extraction Techniques. In *Proceedings of the 2nd International Conference on Web Intelligence, Mining and Semantics*, page 19. ACM, 2012.

[88] Bernd Graubner. ICD and OPS. Historical Development and Current Situation. *Bundesgesundheitsblatt-Gesundheitsforschung-Gesundheitsschutz*, 50:932–943, 2007.

[89] Mark A Greenwood, Valentin Tablan, and Diana Maynard. GATE Mımir: Answering Questions Google Can't. In *Proceedings of the 10th International Semantic Web Conference (ISWC2011)*, 2011.

[90] Ralph Grishman and Beth Sundheim. Message Understanding Conference-6: A Brief History. In *COLING*, volume 96, pages 466–471, 1996.

[91] Tudor Groza, Astrand A Grimnes, and Siegfried Handschuh. Reference Information Extraction and Processing Using Random Conditional Fields. *Information Technology and Libraries*, 31(2):6–20, 2012.

[92] Pankaj Gulhane, Rajeev Rastogi, Srinivasan H. Sengamedu, and Ashwin Tengli. Exploiting Content Redundancy for Web Information Extraction. *Proc. VLDB Endow.*, 3:578–587, 2010.

[93] Zhixin Guo and Hai Jin. A Rule-Based Framework of Metadata Extraction from Scientific Papers. In *Distributed Computing and Applications to Business, Engineering and Science (DCABES), 2011 Tenth International Symposium on*, pages 400–404. IEEE, 2011.

[94] Rahul Gupta, Sunita Sarawagi, and Ajit A Diwan. Collective Inference for Extraction MRFs Coupled with Symmetric Clique Potentials. *The Journal of Machine Learning Research*, 11:3097–3135, 2010.

[95] Iryna Gurevych, Max Mühlhäuser, Christof Müller, Jürgen Steimle, Markus Weimer, and Torsten Zesch. Darmstadt Knowledge Processing Repository Based on UIMA. In *Proceedings of the First Workshop on Unstructured Information Management Architecture at Biannual Conference of the Society for Computational Linguistics and Language Technology*, Tübingen, Germany, April 2007.

[96] Marti A. Hearst, ST Dumais, E Osman, John Platt, and Bernhard Scholkopf. Support Vector Machines. *Intelligent Systems and their Applications, IEEE*, 13(4):18–28, 1998.

[97] Erik Hetzner. A Simple Method for Citation Metadata Extraction using Hidden Markov Models. In *Proceedings of the 8th ACM/IEEE-CS joint Conference on Digital libraries*, pages 280–284. ACM, 2008.

[98] Jerry R. Hobbs, Douglas Appelt, John Bear, David Israel, Megumi Kameyama, and Mabry Tyson. FASTUS: A System for Extracting Information from Text. In *Proceedings of the Workshop on Human Language Technology*, HLT '93, pages 133–137, Stroudsburg, PA, USA, 1993. Association for Computational Linguistics.

[99] Alexander Hoernlein, Marianus Ifland, Peter Kluegl, and Frank Puppe. Konzeption und Evaluation eines fallbasierten Trainingssystems im universitätsweiten Einsatz (CaseTrain). *GMS Medizinische Informatik, Biometrie und Epidemiologie, GMS Med Inform Biom Epidemiol*, 5(1), 2008.

[100] Ching Hoi Andy Hong, Jesse Prabawa Gozali, and Min-Yen Kan. FireCite: Lightweight Real-time Reference String Extraction from Webpages. In *Proceedings of the 2009 Workshop on Text and Citation Analysis for Scholarly Digital Libraries*, pages 71–79. Association for Computational Linguistics, 2009.

[101] Andreas Hotho, Robert Jaeschke, Christoph Schmitz, and Gerd Stumme. BibSonomy: A Social Bookmark and Publication Sharing System. In *Proceedings of the Conceptual Structures Tool Interoperability Workshop at the 14th International Conference on Conceptual Structures*, pages 87–102. Aalborg University Press, 2006.

[102] ScottB. Huffman. Learning Information Extraction Patterns from Examples. In Stefan Wermter, Ellen Riloff, and Gabriele Scheler, editors, *Connectionist, Statistical and Symbolic Approaches to Learning for Natural Language Processing*, volume 1040 of *Lecture Notes in Computer Science*, pages 246–260. Springer Berlin Heidelberg, 1996.

[103] Betsy L Humphreys, Donald AB Lindberg, Harold M Schoolman, and G Octo Barnett. The Unified Medical Language System An Informatics Research Collaboration. *Journal of the American Medical Informatics Association*, 5(1):1–11, 1998.

[104] Dat T Huynh and Wen Hua. Self-supervised Learning Approach for Extracting Citation Information on the Web. In *Web Technologies and Applications*, pages 719–726. Springer, 2012.

[105] Wouter IJntema, Jordy Sangers, Frederik Hogenboom, and Flavius Frasincar. A Lexico-semantic Pattern Language for Learning Ontology Instances from Text. *Web Semantics: Science, Services and Agents on the World Wide Web*, 15:37–50, September 2012.

[106] Michal Jacovi, Vladimir Soroka, Gail Gilboa-Freedman, Sigalit Ur, Elad Shahar, and Natalia Marmasse. The Chasms of CSCW: a Citation Graph Analysis of the CSCW Conference. In *Proceedings of the 2006 20th anniversary Conference on Computer supported cooperative work*, pages 289–298. ACM, 2006.

[107] David Janzen and Hossein Saiedian. Test-Driven Development: Concepts, Taxonomy, and Future Direction. *Computer*, 38(9):43–50, 2005.

[108] Michael Jewell. Paracite: An Overview. published online, 2000. http://paracite.eprints.org/docs/overview.html.

[109] A Jordan. On Discriminative vs. Generative Classifiers: A Comparison of Logistic Regression and Naive Bayes. *Advances in neural information processing systems*, 14:841, 2002.

[110] Tomasz Kaczmarek, Marek Kowalkiewicz, and Jakub Piskorski. Information Extraction from CV. In *8th International Conference on Business Information Systems*, 2005.

[111] Yoshinobu Kano, William A Baumgartner, Luke McCrohon, Sophia Ananiadou, K Bretonnel Cohen, Lawrence Hunter, and Jun'ichi Tsujii. U-Compare: Share and Compare Text Mining Tools with UIMA. *Bioinformatics*, 25(15):1997–1998, 2009.

[112] Roman Kern and Stefan Klampfl. Extraction of References Using Layout and Formatting Information from Scientific Articles. *D-Lib Magazine*, 19(9/10), 2013.

[113] Sanjeet Khaitan, Ganesh Ramakrishnan, Sachindra Joshi, and Anup Chalamalla. RAD: A Scalable Framework for Annotator Development. In Gustavo Alonso, José A. Blakeley, and Arbee L. P. Chen, editors, *ICDE*, pages 1624–1627. IEEE, 2008.

[114] Jun-Tae Kim and D.I. Moldovan. Acquisition of Linguistic Patterns for Knowledge-based Information Extraction. *Knowledge and Data Engineering, IEEE Transactions on*, 7(5):713–724, 1995.

[115] Young-Min Kim, Patrice Bellot, Elodie Faath, and Marin Dacos. Annotated Bibliographical Reference Corpora in Digital Humanities. In Nicoletta Calzolari (Conference Chair), Khalid Choukri, Thierry Declerck, Mehmet Uğur Doğan, Bente Maegaard, Joseph Mariani, Jan Odijk, and Stelios Piperidis, editors, *Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC'12)*, Istanbul, Turkey, may 2012. European Language Resources Association (ELRA).

[116] Young-Min Kim, Patrice Bellot, Jade Tavernier, Elodie Faath, and Marin Dacos. Evaluation of BILBO Reference Parsing in Digital Humanities via a Comparison of Different Tools. In *Proceedings of the 2012 ACM symposium on Document engineering*, pages 209–212. ACM, 2012.

[117] W. Klösgen. Explora: A Multipattern and Multistrategy Discovery Assistant. In U. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy, editors, *Advances in Knowledge Discovery and Data Mining*, pages 249–271. AAAI Press, 1996.

[118] Peter Kluegl, Martin Atzmueller, Tobias Hermann, and Frank Puppe. A Framework for Semi-Automatic Development of Rule-based Information Extraction Applications. In Melanie Hartmann und Frederik Janssen, editor, *Proc. LWA 2009 (KDML - Special Track on Knowledge Discovery and Machine Learning)*, pages 56–59, 2009.

[119] Peter Kluegl, Martin Atzmueller, and Frank Puppe. Integrating the Rule-Based IE Component TextMarker into UIMA. In Joachim Baumeister and Martin Atzmueller, editors, *LWA-2008 (Special Track on Information Retrieval)*, pages 73–77, 2008.

[120] Peter Kluegl, Martin Atzmueller, and Frank Puppe. Test-Driven Development of Complex Information Extraction Systems using TextMarker. In Grzegorz J. Naplepa and Joachim Baumeister, editors, *4th International Workshop on Knowledge Engineering and Software Engineering (KESE 2008),31th German Conference on Artificial Intelligence (KI-2008)*, pages 19–30, 2008.

[121] Peter Kluegl, Martin Atzmueller, and Frank Puppe. Meta-level Information Extraction. In Bärbel Mertsching, Marcus Hund, and Muhammad Zaheer Aziz, editors, *KI*, volume 5803 of *Lecture Notes in Computer Science*, pages 233–240. Springer, 2009.

[122] Peter Kluegl, Martin Atzmueller, and Frank Puppe. TextMarker: A Tool for Rule-Based Information Extraction. In Christian Chiarcos, Richard Eckart de Castilho, and Manfred Stede, editors, *Proceedings of the Biennial GSCL Conference 2009, 2nd UIMA@GSCL Workshop*, pages 233–240. Gunter Narr Verlag, 2009.

[123] Peter Kluegl, Andreas Hotho, and Frank Puppe. Local Adaptive Extraction of References. In Rüdiger Dillmann, Jürgen Beyerer, Uwe D. Hanebeck, and Tanja Schultz, editors, *KI 2010: Advances in Artificial Intelligence, 33rd Annual German Conference on AI*, LNAI 6359, pages 40–47. Springer, 2010.

[124] Peter Kluegl and Martin Toepfer. Informationsextraktion. *Informatik-Spektrum*, 37(2):132–135, 2014.

[125] Peter Kluegl, Martin Toepfer, Philip-Daniel Beck, Georg Fette, and Frank Puppe. UIMA Ruta: Rapid Development of Rule-based Information Extraction Applications. *Natural Language Engineering*, FirstView:1–40, 12 2014.

[126] Peter Kluegl, Martin Toepfer, Philip-Daniel Beck, Georg Fette, and Frank Puppe. UIMA Ruta Workbench: Rule-based Text Annotation. In *System Demonstration,COLING*, 2014. accepted.

[127] Peter Kluegl, Martin Toepfer, Florian Lemmerich, Andreas Hotho, and Frank Puppe. Collective Information Extraction with Context-Specific Consistencies. In Peter A. Flach, Tijl De Bie, and Nello Cristianini, editors, *ECML/PKDD (1)*, volume 7523 of *Lecture Notes in Computer Science*, pages 728–743. Springer, 2012.

[128] Peter Kluegl, Martin Toepfer, Florian Lemmerich, Andreas Hotho, and Frank Puppe. Stacked Conditional Random Fields Exploiting Structural Consistencies. In Pedro Latorre Carmona, J. Salvador Sanchez, and Ana Fred, editors, *Proceedings of 1st International*

Conference on Pattern Recognition Applications and Methods (ICPRAM), pages 240–248, Vilamoura, Algarve, Portugal, February 2012. SciTePress.

[129] Peter Kluegl, Martin Toepfer, Florian Lemmerich, Andreas Hotho, and Frank Puppe. Exploiting Structural Consistencies with Stacked Conditional Random Fields. *Mathematical Methodologies in Pattern Recognition and Machine Learning, Springer Proceedings in Mathematics and Statistics*, 30:111–125, 2013.

[130] Sunil Kumar Kopparapu. Automatic Extraction of Usable Information from Unstructured Resumes to Aid Search. In *Progress in Informatics and Computing (PIC), 2010 IEEE International Conference on*, volume 1, pages 99–103. IEEE, 2010.

[131] Zhenzhen Kou and William W. Cohen. Stacked Graphical Models for Efficient Inference in Markov Random Fields. In *Proceedings of the 2007 SIAM Int. Conf. on Data Mining*, 2007.

[132] Hans-Ulrich Krieger, Witold Drozdzynski, Jakub Piskorski, Ulrich Schäfer, and Feiyu Xu. A Bag of Useful Techniques for Unification-Based Finite-State Transducers. In Ernst Buchberger, editor, *Proceedings of 7th KONVENS*, pages 105–112, 9 2004.

[133] Hans-Ulrich Krieger and Ulrich Schäfer. TDL - A Type Description Language for Constraint-Based Grammars. In ACL, editor, *Proceedings of the 15th International Conference on Computational Linguistics (COLING '94), August 5-9*, volume 2, pages 893–899, Kyoto, Japan, 1994.

[134] Vijay Krishnan and Christopher D. Manning. An Effective two-stage Model for Exploiting non-local Dependencies in Named Entity Recognition. In *Proc. of the 21st Int. Conf. on Computational Linguistics and the 44th Annual Meeting of the ACL*, ACL-44, pages 1121–1128, Stroudsburg, PA, USA, 2006. ACL.

[135] Frank R Kschischang, Brendan J Frey, and H-A Loeliger. Factor Graphs and the Sum-product Algorithm. *Information Theory, IEEE Transactions on*, 47(2):498–519, 2001.

[136] N. Kushmerick, D. Weld, and B. Doorenbos. Wrapper Induction for Information Extraction. In *Proc. IJC Artificial Intelligence*, 1997.

[137] J. Lafferty, A. McCallum, and F. Pereira. Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data. *Proc. 18th International Conf. on Machine Learning*, pages 282–289, 2001.

[138] Steve Lawrence, C Lee Giles, and Kurt D Bollacker. Autonomous Citation Matching. In *Proceedings of the third annual Conference on Autonomous Agents*, pages 392–393. ACM, 1999.

[139] Yaoyong Li, Kalina Bontcheva, and Hamish Cunningham. Adapting SVM for Data Sparseness and Imbalance: A Case Study on Information Extraction. *Natural Language Engineering*, 15(2):241–271, 2009.

[140] Ying Li, Sharon Lipsky Gorman, and Noemie Elhadad. Section Classification in Clinical Notes using Supervised Hidden Markov Model. In Tiffany C. Veinot, Ümit V. Çatalyürek, Gang Luo, Henrique Andrade, and Neil R. Smalheiser, editors, *IHI*, pages 744–750. ACM, 2010.

[141] Yunyao Li, Laura Chiticariu, Huahai Yang, Frederick R. Reiss, and Arnaldo Carreno-fuentes. WizIE: a Best Practices Guided Development Environment for Information Extraction. In *Proceedings of the ACL 2012 System Demonstrations*, ACL '12, pages 109–114, Stroudsburg, PA, USA, 2012. Association for Computational Linguistics.

[142] Mario Lipinski, Kevin Yao, Corinna Breitinger, Joeran Beel, and Bela Gipp. Evaluation of Header Metadata Extraction Approaches and Tools for Scientific PDF Documents. In *Proceedings of the 13th ACM/IEEE-CS Joint Conference on Digital Libraries*, JCDL '13, pages 385–386, New York, NY, USA, 2013. ACM.

[143] Patrice Lopez. Automatic Extraction and Resolution of Bibliographical References in Patent Documents. In *Advances in Multidisciplinary Retrieval*, pages 120–135. Springer, 2010.

[144] Ashwin Machanavajjhala, Arun Shankar Iyer, Philip Bohannon, and Srujana Merugu. Collective Extraction from Heterogeneous Web Lists. In *Proceedings of the Fourth ACM International Conference on Web Search and Data Mining*, WSDM '11, pages 445–454, New York, NY, USA, 2011. ACM.

[145] Sumit Maheshwari, Abhishek Sainani, and P Krishna Reddy. An Approach to Extract Special Skills to Improve the Performance of Resume SSelection. In *Databases in Networked Information Systems*, pages 256–273. Springer, 2010.

[146] Gideon S. Mann and Andrew McCallum. Generalized Expectation Criteria for Semi-Supervised Learning with Weakly Labeled Data. *J. Mach. Learn. Res.*, 11:955–984, 2010.

[147] Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. The Stanford CoreNLP Natural Language Processing Toolkit. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 55–60, 2014.

[148] E. Michael Maximilien and Laurie Williams. Assessing Test-Driven Development at IBM. In *ICSE '03: Proceedings of the 25th International Conference on Software Engineering*, pages 564–569, Washington, DC, USA, 2003. IEEE Computer Society.

[149] AT Mc Cray. A. The Unified Medical Language System. *Meth Inf Med*, 34:281–291, 1993.

[150] Andrew McCallum. Efficiently Inducing Features of Conditional Random Fields. In *Nineteenth Conference on Uncertainty in Artificial Intelligence (UAI03)*, 2003.

[151] Andrew McCallum, Dayne Freitag, and Fernando CN Pereira. Maximum Entropy Markov Models for Information Extraction and Segmentation. In *ICML*, pages 591–598, 2000.

[152] Andrew McCallum, Karl Schultz, and Sameer Singh. Factorie: Probabilistic Programming via Imperatively Defined Factor Graphs. In *Advances in Neural Information Processing Systems*, pages 1249–1257, 2009.

[153] Andrew Kachites Mccallum, Kamal Nigam, Jason Rennie, and Kristie Seymore. Automating the Construction of Internet Portals with Machine Learning. *Information Retrieval Journal*, 3:127–163, 2000.

[154] Daniel D. McCracken and Edwin D. Reilly. Backus-Naur form (BNF). In *Encyclopedia of Computer Science*, pages 129–131. John Wiley and Sons Ltd., Chichester, UK.

[155] Ion Muslea, Steven Minton, and CraigA. Knoblock. Hierarchical Wrapper Induction for Semistructured Information Sources. *Autonomous Agents and Multi-Agent Systems*, 4(1-2):93–114, 2001.

[156] Philip Ogren and Steven Bethard. Building Test Suites for UIMA Components. In *Proceedings of the Workshop on Software Engineering, Testing, and Quality Assurance for Natural Language Processing (SETQA-NLP 2009)*, pages 1–4, Boulder, Colorado, June 2009. Association for Computational Linguistics.

[157] Manabu Ohta, Daiki Arauchi, Atsuhiro Takasu, and Jun Adachi. CRF-based Bibliography Extraction from Reference Strings Focusing on Various Token Granularities. In *Document Analysis Systems (DAS), 2012 10th IAPR International Workshop on*, pages 276–281. IEEE, 2012.

[158] Stefan Olbrecht. Regelbasierte Named Entity Recognition in literarischen Texten (in German). Bachelor thesis, University of Würzburg, 2014.

[159] Sung Hee Park, Roger W Ehrich, and Edward A Fox. A Hybrid Two-stage Approach for Discipline-independent Canonical Representation Extraction from References. In *Proceedings of the 12th ACM/IEEE-CS joint Conference on Digital Libraries*, pages 285–294. ACM, 2012.

[160] Fuchun Peng and Andrew McCallum. Accurate Information Extraction from Research Papers using Conditional Random Fields. In *HLT-NAACL*, pages 329–336, 2004.

[161] Jakub Piskorski. Finite-State Machine Toolkit. Technical Report RR-02-04, DFKI, Saarbrücken, 2002.

[162] Jakub Piskorski and Roman Yangarber. Information Extraction: Past, Present and Future. In Thierry Poibeau, Horacio Saggion, Jakub Piskorski, and Roman Yangarber, editors, *Multi-source, Multilingual Information Extraction and Summarization*, Theory and Applications of Natural Language Processing, pages 23–49. Springer Berlin Heidelberg, 2013.

[163] Hoifung Poon and Pedro Domingos. Joint Inference in Information Extraction. In *AAAI'07: Proceedings of the 22nd National Conference on Artificial intelligence*, pages 913–918. AAAI Press, 2007.

[164] Brett Powley and Robert Dale. Evidence-based Information Extraction for High Accuracy Citation and Author Name Identification. In *Large Scale Semantic Access to Content (Text, Image, Video, and Sound)*, pages 618–632, 2007.

[165] John Ross Quinlan. *C4.5: Programs for Machine Learning*, volume 1. Morgan kaufmann, 1993.

[166] Lawrence Rabiner. A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.

[167] Lawrence Rabiner and Biing-Hwang Juang. An Introduction to Hidden Markov Models. *ASSP Magazine, IEEE*, 3(1):4–16, 1986.

[168] Ganesh Ramakrishnan, Sreeram Balakrishnan, and Sachindra Joshi. Entity Annotation based on Inverse Index Operations. In Dan Jurafsky and Éric Gaussier, editors, *EMNLP*, pages 492–500. ACL, 2006.

[169] Frederick Reiss, Sriram Raghavan, Rajasekar Krishnamurthy, Huaiyu Zhu, and Shivakumar Vaithyanathan. An Algebraic Approach to Rule-Based Information Extraction. In *Proceedings of the 2008 IEEE 24th International Conference on Data Engineering*, ICDE '08, pages 933–942, Washington, DC, USA, 2008. IEEE Computer Society.

[170] Matthew Richardson and Pedro Domingos. Markov Logic Networks. *Machine Learning*, 62(1-2):107–136, 2006.

[171] Ellen Riloff. Automatically Constructing a Dictionary for Information Extraction Tasks. In *Proceedings of the Eleventh National Conference on Artificial Intelligence*, AAAI'93, pages 811–816. AAAI Press, 1993.

[172] Ellen Riloff and William Phillips. An Introduction to the Sundance and Autoslog systems. Technical report, University of Utah, 2004.

[173] Irina Rish. An Empirical Study of the Naive Bayes Classifier. In *IJCAI 2001 workshop on empirical methods in artificial intelligence*, volume 3, pages 41–46, 2001.

[174] Khashayar Rohanimanesh, Kedar Bellare, Aron Culotta, Andrew McCallum, and Michael L Wick. SampleRank: Training Factor Graphs with Atomic Gradients. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 777–784, 2011.

[175] Alexander M. Rush, Roi Reichart, Michael Collins, and Amir Globerson. Improved Parsing and POS Tagging Using Inter-sentence Consistency Constraints. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, EMNLP-CoNLL '12, pages 1434–1444, Stroudsburg, PA, USA, 2012. Association for Computational Linguistics.

[176] Sunita Sarawagi. Information Extraction. *Found. Trends databases*, 1(3):261–377, March 2008.

[177] Guido Sautter and Klemens Böhm. Improved Bibliographic Reference Parsing based on Repeated Patterns. In Panayiotis Zaphiris, George Buchanan, Edie Rasmussen, and Fernando Loizides, editors, *Theory and Practice of Digital Libraries*, volume 7489 of *Lecture Notes in Computer Science*, pages 370–382. Springer Berlin Heidelberg, 2012.

[178] Guido Sautter and Klemens Böhm. Improved Bibliographic Reference Parsing based on Repeated Patterns. *International Journal on Digital Libraries*, 14(1-2):59–80, 2014.

[179] Guergana K. Savova, James J. Masanz, Philip V. Ogren, Jiaping Zheng, Sunghwan Sohn, Karin C. Kipper-Schuler, and Christopher G. Chute. Mayo clinical Text Analysis and Knowledge Extraction System (cTAKES): Architecture, Component Evaluation and Applications. *Journal of the American Medical Informatics Association : JAMIA*, 17(5):507–513, September 2010.

[180] Bernhard Schölkopf and Alexander J Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. MIT press, 2002.

[181] Kristie Seymore, Andrew McCallum, and Roni Rosenfeld. Learning Hidden Markov Model Structure for Information Extraction. In *AAAI-99 Workshop on Machine Learning for Information Extraction*, pages 37–42, 1999.

[182] Warren Shen, AnHai Doan, Jeffrey F. Naughton, and Raghu Ramakrishnan. Declarative Information Extraction Using Datalog with Embedded Extraction Predicates. In *Proceedings of the 33rd International Conference on Very Large Data Bases*, VLDB '07, pages 1033–1044. VLDB Endowment, 2007.

[183] Sameer Singh, Karl Schultz, and Andrew McCallum. Bi-directional Joint Inference for Entity Resolution and Segmentation Using Imperatively-Defined Factor Graphs. In *Proceedings of the European Conference on Machine Learning and Knowledge Discovery in Databases: Part II*, ECML PKDD '09, pages 414–429. Springer-Verlag, 2009.

[184] Seán Slattery and Tom Mitchell. Discovering Test Set Regularities in Relational Domains. In *ICML*, pages 895–902, 2000.

[185] S. G. Soderland. Learning Text Analysis Rules for Domain-specific Natural Language Processing. Technical report, University of Massachusetts, Amherst, MA, USA, 1996.

[186] Stephen Soderland, Claire Cardie, and Raymond Mooney. Learning Information Extraction Rules for Semi-Structured and Free Text. In *Machine Learning*, volume 34, pages 233–272, 1999.

[187] Jan Stadermann, Stephan Symons, and Ingo Thon. Extracting Hierarchical Data Points and Tables from Scanned Contracts. In Peter Klügl, Richard Eckart de Castilho, and Katrin Tomanek, editors, *UIMA@GSCL*, volume 1038 of *CEUR Workshop Proceedings*, pages 50–57. CEUR-WS.org, 2013.

[188] Charles Sutton. GRMM: GRaphical Models in Mallet, 2006.

[189] Charles Sutton and Andrew McCallum. Collective Segmentation and Labeling of Distant Entities in Information Extraction. In *ICML Workshop on Statistical Relational Learning and Its Connections to Other Fields*, 2004.

[190] Charles Sutton and Andrew McCallum. An Introduction to Conditional Random Fields for Relational Learning. *Introduction to statistical relational learning*, pages 93–128, 2006.

[191] Charles Sutton and Andrew McCallum. An Introduction to Conditional Random Fields. *Foundations and Trends® in Machine Leanring*, 4(4), 2012.

[192] Michael Tepper, Daniel Capurro, Fei Xia, Lucy Vanderwende, and Meliha Yetisgen-Yildiz. Statistical Section Segmentation in Free-Text Clinical Records. In Nicoletta Calzolari (Conference Chair), Khalid Choukri, Thierry Declerck, Mehmet Uğur Doğan, Bente Maegaard, Joseph Mariani, Jan Odijk, and Stelios Piperidis, editors, *Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC'12)*, Istanbul, Turkey, may 2012. European Language Resources Association (ELRA).

[193] Erik F. Tjong Kim Sang. Introduction to the CoNLL-2002 Shared Task: Language-independent Named Entity Recognition. In *Proceedings of the 6th Conference on Natural Language Learning - Volume 20*, COLING-02, pages 1–4, Stroudsburg, PA, USA, 2002. Association for Computational Linguistics.

[194] Martin Toepfer, Peter Kluegl, Andreas Hotho, and Frank Puppe. Conditional Random Fields For Local Adaptive Reference Extraction. In Martin Atzmüller, Dominik Benz, Andreas Hotho, and Gerd Stumme, editors, *Proceedings of LWA2010 - Workshop-Woche: Lernen, Wissen & Adaptivitaet*, Kassel, Germany, 2010.

[195] Martin Toepfer, Peter Kluegl, Andreas Hotho, and Frank Puppe. Segmentation of References with Skip-Chain Conditional Random Fields for Consistent Label Transitions. In *Workshop Notes of the LWA 2011 - Learning, Knowledge, Adaptation*, 2011.

[196] Jordi Turmo, Alicia Ageno, and Neus Català. Adaptive Information Extraction. *ACM Comput. Surv.*, 38(2), July 2006.

[197] Andrew J Viterbi. Error Bounds for Convolutional Codes and an Asymptotically Optimum Decoding Algorithm. *Information Theory, IEEE Transactions*, 13(2):260–269, 1967.

[198] Patrick von Schoen. Automatische Aufbereitung von Arztbriefen für Trainingsfälle mittels Anonymisierung und Terminologie-Matching. Diploma thesis, University of Würzburg, 2006.

[199] Jakob Voss, Andreas Hotho, and Robert Jaeschke. Mapping Bibliographic Records with Bibliographic Hash Keys. In Rainer Kuhlen, editor, *Information: Droge, Ware oder Commons?*, Proceedings of the ISI. Hochschulverband Informationswissenschaft, Verlag Werner Huelsbusch, 2009.

[200] Martin J. Wainwright, Tommi Jaakkola, and Alan S. Willsky. Tree-based Reparameterization for Approximate Inference on Loopy Graphs. In *NIPS*, pages 1001–1008, 2001.

[201] Hanna M Wallach. Conditional Random Fields: An Introduction. *Technical Reports (CIS)*, page 22, 2004.

[202] Michael L. Wick, Aron Culotta, Khashayar Rohanimanesh, and Andrew McCallum. An Entity Based Model for Coreference Resolution. In *SDM*, pages 365–376. SIAM, 2009.

[203] Andreas Wittek, Martin Toepfer, Georg Fette, Peter Kluegl, and Frank Puppe. Constraint-driven Evaluation in UIMA Ruta. In Peter Kluegl, Richard Eckart de Castilho, and Katrin Tomanek, editors, *UIMA@GSCL*, volume 1038 of *CEUR Workshop Proceedings*, pages 58–65. CEUR-WS.org, 2013.

[204] Tak-Lam Wong and Wai Lam. Adapting Web Information Extraction Knowledge via Mining Site-invariant and Site-dependent Features. *ACM Transactions on Internet Technology (TOIT)*, 7(1):6, 2007.

[205] Ying Xu, Mi-Young Kim, Kevin Quinn, Randy Goebel, and Denilson Barbosa. Open Information Extraction with Tree Kernels. In *HLT-NAACL*, pages 868–877, 2013.

[206] Huahai Yang, Daina Pupons-Wickham, Laura Chiticariu, Yunyao Li, Benjamin Nguyen, and Arnaldo Carreno-Fuentes. I Can Do Text Analytics!: Designing Development Tools for Novice Developers. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '13, pages 1599–1608, New York, NY, USA, 2013. ACM.

[207] Jiang-Ming Yang, Rui Cai, Yida Wang, Jun Zhu, Lei Zhang, and Wei-Ying Ma. Incorporating Site-level Knowledge to Extract Structured Data from Web Forums. In *Proceedings of the 18th International Conference on World Wide Web*, pages 181–190. ACM, 2009.

[208] Jonathan S Yedidia, William T Freeman, and Yair Weiss. Understanding Belief Propagation and its Generalizations. *Exploring artificial intelligence in the new millennium*, 8:236–239, 2003.

[209] Kun Yu, Gang Guan, and Ming Zhou. Resume Information Extraction with Cascaded Hybrid Model. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, ACL '05, pages 499–506, Stroudsburg, PA, USA, 2005. Association for Computational Linguistics.

[210] Qing Zhang, Yong-Gang Cao, and Hong Yu. Parsing Citations in Biomedical Articles using Conditional Random Fields. *Computers in biology and medicine*, 41(4):190–194, 2011.

[211] Jie Zou, Daniel Le, and George R Thoma. Locating and Parsing Bibliographic References in HTML Medical Articles. *International Journal on Document Analysis and Recognition (IJDAR)*, 13(2):107–119, 2010.

Information extraction is widely used to identify well-defined entities and relations in unstructured data. Interesting entities are often consistently structured within a certain context, especially in semi-structured texts. However, their actual composition varies and is possibly inconsistent among different contexts. Information extraction models stay behind their potential and return inferior results if they do not consider these consistencies during processing. This work presents a selection of practical and novel approaches for exploiting these context-specific consistencies in information extraction tasks. The approaches direct their attention not only to one technique, but are based on handcrafted rules as well as probabilistic models.

ISBN 978-3-95826-018-4
9 783958 260184

Würzburg University Press