Applied Intelligence (2023) 53:9513–9530
https://doi.org/10.1007/s10489-022-03957-8

KIETA: Key-insight extraction from scientific tables

Sebastian Kempf[1] · **Markus Krug**[1] · **Frank Puppe**[1]

Accepted: 2 July 2022 / Published online: 9 August 2022

Abstract
An important but very time consuming part of the research process is literature review. An already large and nevertheless growing ground set of publications as well as a steadily increasing publication rate continue to worsen the situation. Consequently, automating this task as far as possible is desirable. Experimental results of systems are key-insights of high importance during literature review and usually represented in form of tables. Our pipeline *KIETA* exploits these tables to contribute to the endeavor of automation by extracting them and their contained knowledge from scientific publications. The pipeline is split into multiple steps to guarantee modularity as well as analyzability, and agnosticim regarding the specific scientific domain up until the knowledge extraction step, which is based upon an ontology. Additionally, a dataset of corresponding articles has been manually annotated with information regarding table and knowledge extraction. Experiments show promising results that signal the possibility of an automated system, while also indicating limits of extracting knowledge from tables without any context.

Keywords Table extraction · Table understanding · Ontology · Key-insight extraction · Information extraction

1 Introduction

An important part of research is literature review, consisting of selecting, reading and comparing scientific works that correspond to a given research domain. This leads to some problems, because every publication increases the amount of literature to be reviewed. Relatively new and active research fields ten to have a high publication rate, meaning more papers are published in a smaller amount of time. This problem is exemplified by the domain of computer science for which the number of submissions per month on arxiv.org grew from under 2500 per month in 2017 to close to 5000 per month in 2020. As a result the workload of literature review for researchers is increasing steadily. Automating parts of the literature review process would therefore free up time spent reviewing and increase the overall quality as the chances of important publications being missed are lowered.

While current methods of literature review assistance or automation like keyword searches allow narrowing down the set of papers to be considered, the results of these papers still have to be manually compared in order to get an idea of their content and their relative performance. The key during this process is extracting key-insights, which are pieces of information relevant to a researcher. For example a common question within the community of machine learning is "What is the current state-of-the-art on a given dataset?". Here, the relevant key-insights concern experimental results on some dataset. Other common examples of key-insights are the task or methodology. However, there are efforts to create so called leaderboards, either manually maintained or supported by algorithms, to simplify answering the first question concerning the current state-of-the-art. Examples for these leaderboards are paperswithcode[1] and nlp-progress[2]. These list the best experimental results of papers as tuples $(task, dataset, metric, value)$. However, a significant amount of information is lost as all other experimental results contained within the paper are discarded. This data is useful for further statistical analysis, or for

✉ Sebastian Kempf
sebastian.kempf@informatik.uni-wuerzburg.de

Markus Krug
markus.krug@informatik.uni-wuerzburg.de

Frank Puppe
frank.puppe@informatik.uni-wuerzburg.de

[1] Chair of Computer Science VI, University of Würzburg, Sanderring 2, Würzburg, 97070, Bavaria, Germany

[1] https://paperswithcode.com/
[2] https://nlpprogress.com/

example checking the results were reported correctly. This contribution is part of a bigger research effort to develop a pipeline that can extract key-insights from scientific publications. The current focus is knowledge concerning experimental results, which are usually represented in form of tables. Consequently, this approach concentrates on the automated extraction of tables and subsequent generation of included knowledge. This knowledge is defined using an ontology developed within this contribution. While a general ontology describing all possible scientific results would be preferable, their complex nature makes the development very challenging. In order to demonstrate the potential of this task and the applicability of this pipeline, it is currently necessary to limit the scope to a single domain. The domain of coreference resolution has been arbitrarily chosen for the purposes of this contribution. Any other research domain containing experimental results could have been used. We developed *KIETA*, a pipeline to extract tables from scientific documents and subsequently extract key-insights that are defined in an ontology. To this end, a rule- and heuristic-based modular pipeline is created that localizes tables within non-scan PDF documents. The pipeline builds a graph-based table model based on [8] combining structure and semantic meaning in a single model. An ontology is defined in order to formalize the meaning of "experimental result", which is the key-insight that has been chosen as an example. A dataset containing 208 tables from 49 documents is created to serve as the basis of our experiments. Before the construction of this dataset, a two-dimensional categorization of over 750 tables from over 200 documents has been conducted to analyze the real-world environment. The tables in the dataset have been chosen to form a valid sample emulating the real-world distribution as close as possible. Dataset tables are annotated with information concerning every step of the table extraction process relevant for the evaluation. A subset of these tables was also annotated with information regarding the subsequent extraction of knowledge. To the best of our knowledge, KIETA is the first system combining localization, the creation of a multi-faceted table model and subsequent key-insight extraction to a single pipeline. Experiments show that it outperforms popular open-source tools regarding detecting and analyzing tables.

2 Background

This pipeline as well as the evaluation process are based on the table model developed by [8]. A set of cell identifiers T is paired with different restrictions to form the four components of this model. The *Physical Model* represents the typical two-dimensional grid structure of cells by assigning each cell identifier a pair of coordinates within that grid $c = (t \in T, x_1, y_1, x_2, y_2, \text{string})$. Tables usually consist of two kinds of cells: labels and entries. This is described by the *Functional Model* composed of two subsets $A \subset T$, $D \subset T$, where A is the set of row A_r and column A_c labels (access cells, headers) and D the set of entries (data cells). Although the distinction between row and column labels is usually clear, it fails when the stub head is considered. The correct association of the stub head can usually only be achieved using semantic knowledge. For the purposes of the table model within this contribution, the stub head is defined as a column label for the functional evaluation, but otherwise not considered. Although the *Semantic Model* contains many different aspects, this work is focused on the semantics of relations, meaning the interpretation of the table as a database table. Consequently, the model is seen as a mapping of labels to data entries. As an example, the previously described model is applied to Table 1. Note that the stub head is mostly ignored within this contribution, although it is included in this example.

Technically, cell identifiers $c_1, \ldots, c_n$ would have to be used in every example, but for better comprehension the cell content will be used as identifier. This results in the following:

$$
\begin{aligned}
T_{physical} = \; & \{(\text{c1, 2,1,3,1, Weather}),\\
& (\text{c2, 1,1,1,2, Day}),\\
& (\text{c3, 2,2,2,2, Sun}), \ldots\},\\
T_{functional} = \; & \{\{\text{Weather, Day, Sun, Rain, Monday, Friday}\},\\
& \{\text{5h, 0h, 0ml, 10ml}\}\},\\
T_{semantic} = \; & \{((\text{Weather,Sun}), (\text{Day,Monday}), \text{5h}),\\
& (\text{Weather,Sun}), (\text{Day,Friday}), \text{0h}),\\
& (\text{Weather,Rain}), (\text{Day,Monday}), \text{0ml}),\\
& ((\text{Weather,Rain}), (\text{Day,Friday}), \text{10ml})\}.
\end{aligned}
$$

3 Related work

This contribution is related to a number of different fields of study: *table extraction (TE)*, *key-insight extraction (KIE)* and *semantic table interpretation (STI)*. TE means detecting the presence of a table within a given medium (detection),

Table 1 Example table with grid cell coordinates for the application of the table model of [8]

	1	2	3
1		Weather	
2	Day	Sun	Rain
3	Monday	5h	0ml
4	Friday	0h	10ml

defining its boundary (localization) and recovering its structure (recognition). Each is a separate research field being tackled by different existing systems. However, preliminary experiments showed that these systems exhibit issues for the given use case, necessitating the development of a custom variant. KIE concerns the extraction of defined pieces of knowledge from a given medium (e.g. text). The definition of knowledge is a research field of its own and is only briefly discussed here. The focus lies on extracting homogeneous data in order to create a unified and queryable datastructure. The goal of STI is annotating a table with semantic information, i.e. semantifying web tables with as much knowledge as possible. The process relies upon structured tables and usually involves recognizing Named Entities using a knowledge base or graph. As a complete overview of all related work is difficult, the following describes related work employed within our experiments as well as approaches tackling similar problems.

The popular open-source Tabula[3] is promising localization and structure recognition of tables within PDF files. While the algorithmic background is not clearly stated, the source code of the table detection algorithm references [13] in which two heuristic methods to detect bordered and borderless tables are developed. Bordered tables are found using the crossing points of horizontal and vertical lines to define rectangular spaces. These are merged or separated according to heuristically derived rules, establishing the exact boundaries of the tables. This approach cannot be used in case of borderless tables, therefore each row of textual elements is scored according to its probability being a table element using heuristic factors like the presence of numbers. High scoring areas are merged to form a rectangular area, which is defined as table. The recognition approach of Tabula relies on vertical/horizontal lines or a distance threshold to find the correct column/row arrangement.

TableSeer [12] is a table extraction (TE) system specifically geared towards scientific publications that is also able to extract table captions. Textual elements within PDF files are extracted and merged into lines based on their position. Using a set of conditions, lines are further aggregated into boxes and categorized using rules based on their font size. Table candidates are generated by keyword matching and checking if the whitespace of this area matches the general structure of a table. Another set of rules divides these table boxes into classes like "caption" and "column label". The system presented within [11] to detect and localize tables is based on the ResNeXt [20] architecture using the Faster R-CNN algorithm [17]. It has been trained on the TableBank dataset that was also developed within the publication.

Perez-Arriaga et al. [15] presented a system with the goal of capturing the ontological concept of scientific publications. Through noun phrase analysis, named entities (NE) are found within tables and annotated by querying DBpedia. Disambiguation is achieved by Latent Semantic Indexing to find the most similar concept. Relationships between these NEs are found within the text of the publication using a confidence score with logistic regression. A formal representation is achieved by a defined ontology. Oelen et al [14] propose a prototype creating knowledge graphs from tables of scientific surveys. After pre-selecting publications based on several defined characteristics of desirable documents and tables, *Tabula* is employed to extract the tables. The performance of *Tabula* has been deemed not satisfactory by the authors, so errors have to be manually corrected. Knowledge gained from the list of references is annotated to the corresponding rows of a survey table, so that this table contains all necessary information to build a knowledge graph. The data contained within the tables is not interpreted.

Considering the extraction of key-insights, the topic of leaderboard creation is closest to the goal of this contribution. Hou et al. [5] model the task of automatically creating a leaderboard using (Task, Dataset, Metric, Score)-tuples (TDMS) as a natural language inference (NLI) task. GROBID [4] is used to extract tables from PDF documents. These tables are paired with additional information, such as the sentences most likely to describe the experimental setup. A transformer model is employed to predict (TDM)-triples on the basis of these data pairs. Additionally, a separately trained model predicts the relationship between the score and a (TDM)-tuple. Only scores in bold font are considered, as these are most likely to be relevant for a leaderboard. In contrast, our contribution considers all experimental and numerical results found within the table scope. The authors of the AxCell system [9] argue that using PDF documents as input, especially for Table Extraction, leads to noise that can be prevented by extracting information directly from the LaTeXsource. Tables are classified as being relevant for future processing using an ULMFiT classifier and each cell is categorized into similar categories. Five different contexts are generated for each cell, describing the semantic context within caption, text fragment, abstract and the whole paper. Finally, metric values and predefined (TDM)-tuples are associated using evidence of the latter found within the cell contexts. Again, only the highest values are kept, all other values are filtered.

4 Description of KIETA

The goal of our still ongoing research is extracting key-insights from scientific publications in addition to making them queryable and automatically processable. Because the amount and density of information contained within them is

[3] https://github.com/tabulapdf/tabula/

high, tables are the starting building block of this endeavor. Figure 1 gives an overview of our pipeline for extracting tables and their knowledge.

The pipeline is built as a modular system where every part can be exchanged, if the output stays the same. This has the advantage that the performance of components can be evaluated separately, enabling a more detailed evaluation overall and easier comparison of different approaches. The propagation of errors especially important in this pipeline, because every step builds upon the output of the previous one, meaning that errors happening early on in the pipeline might cascade in subsequent steps. Therefore, evaluating as many steps as possible is beneficial.

In contrast, end-to-end systems can only be evaluated at the end, knowing neither origin nor propagation of errors. Although arguably error propagation does not exist in end-to-end systems, knowing what exactly has to be changed in order to improve the performance of the system is still difficult. A still unanswered question is why no existing system has been employed for at least parts of the pipeline. The review of existing tools (*Tabula*, *Camelot* and *TableSeer*) also consisted of testing them on a small number of documents. This experiment showed that these tools were not suited for our purposes. The TE tools *Tabula* and *Camelot* detect relatively few tables and those that could be detected are usually not reconstructed correctly. At the same time, no information useful for improving the quality is given. Furthermore, the table caption is not extracted at all. Although the caption is extracted by *TableSeer*, it is unable to consistently detect tables. In contrast to this, *ResNeXt* tends to predict too many tables, while also merging tables on the same level or splitting a single table. Consequently, building a pipeline meeting the requirements described above was necessary. Excluding the KIE step, our pipeline requires no domain-specific knowledge. Therefore, using coreference resolution as base for the knowledge extraction is more or less arbitrary. Developing an ontology modeling results of different domains is intricate. Therefore, the current focus is limited to a single domain. Nonetheless, our pipeline could easily be adapted to work with domains other than coreference resolution, due to its domain-agnosticism. Each page of a given PDF document is preprocessed by converting it into a XML file containing the text and additionally render it as a picture. Afterwards, the presence of tables is detected and their boundary as well as their caption determined. The recognition module constructs a model describing the table on the basis of a graph. This graph represents the table structure as well as the function of each node within the table (e.g. label). Finally, the extraction module extracts key-insights as defined in an ontology. The extraction consists of searching for keyword occurrences of ontology instances within the table graph and exploiting the graph structure to create tuples representing the desired knowledge.

4.1 Preprocessing

In contrast to other works [9, 18], the input is assumed to be a PDF document, meaning the exported output of a document editor like LaTeXor office programs. Although the source of scientific articles is often available for articles on pre-print platforms, PDF documents are considerably more common. The PDF file is converted into a structured XML format using *pdfalto*[4]. This tool automatically merges characters to form words and aggregates them into units like lines and blocks. Because this format contains the characters as written in the PDF stream, the quality of the extracted text is expected to be higher than the output of an OCR engine, which would be necessary if the PDF document was a scan. Furthermore, each textual unit is annotated with positional information that corresponds to their visual location within the PDF page. Additionally, the PDF document is rendered as an image using *pdf2image*[5], because *pdfalto* is (currently) not able to extract elements like vertical and horizontal lines reliably.

4.2 Detection

The distinction between detecting (identifying the presence) and localizing (establishing the boundary) a table is uncommon. Normally, the former is combined with the latter. However within the scope of this publication, these two steps are considered separately. The reason can easily be seen in the following scenario. Two tables are on a given page and the system was able to establish the bounding box of (localize) one table, but failed to do the same for the other table. Given a production environment, the second table would be lost, consequently lowering the extraction quality. However, if the system considered the Detection step separately, the information that there are two tables on the page would not be missed. This information could be used to demand some kind of human input.

A convenient standard of scientific publication is that tables are usually accompanied by captions. These captions follow a structure that could be expressed as a tuple in the form of $(keyword, id, delimiter, string)$. The caption "Table 1: Example table" would translate to (Table 1, :, "Example table"). Ignoring the string for the moment, the triple structure of (keyword, id, delimiter) is very common and easy to recognize. This Detection module exploits this fact by matching each line in the XML file against a list

[4] https://github.com/kermitt2/pdfalto, GPL-2.0 License

[5] https://github.com/Belval/pdf2image, MIT License

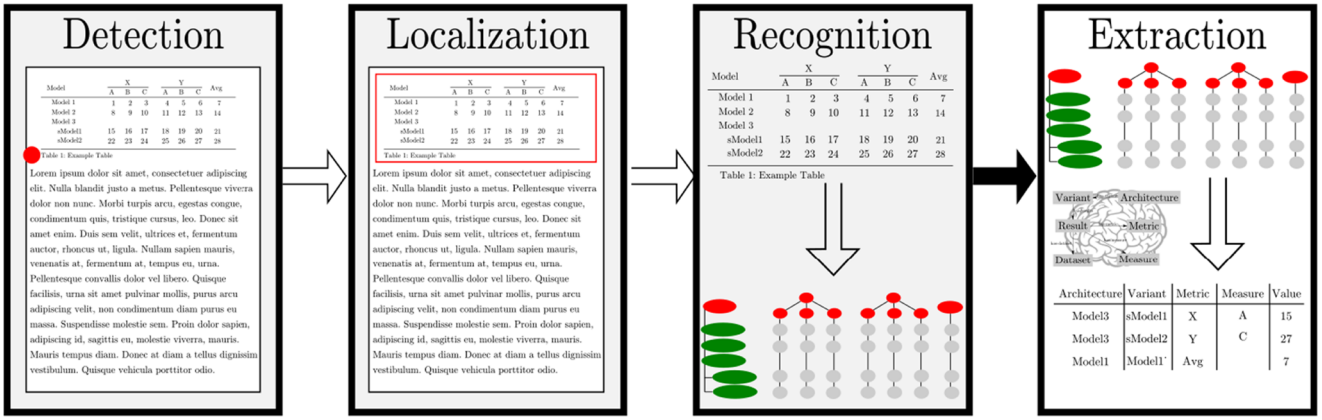

Fig. 1 Schematic overview of *KIETA*

of templates describing different layouts. Since there is a finite number of possible layouts, the list will include all common layouts if unknown templates are simply appended to the list. In order to filter out occurrences of templates within the text, this module relies the on the delimiter being consistent throughout the document, allowing filtering out irregularities. Several heuristics are employed to differentiate caption beginnings from regular text occurrences. They include assumptions like these triples being at the start of a line as well as the capitalization of the first word after the triple.

4.3 Localization

The next step is establishing the actual boundary of the table. This pipeline models two equivalent ways how this boundary can be defined. The first is based on the bounding box, meaning a quadruple $B = (x1, y1, x2, y2)$ defining two points of a rectangle encompassing all elements. The second possibility is a table as a subset of elements within the page $T \subseteq E$. These can be converted into each other, as the bounding box is at least as big as the union of all element bounding boxes and the bounding box automatically defines a set T. Two different approaches have been developed based on these aspects. Both assume that a caption is either written below or above a table, meaning that side-captions can not be detected. Furthermore, a table is only associated with a single caption. This allows reducing the search space, because if another caption is encountered, a table situated "behind" this other caption makes an association between the table and the current caption impossible. Both modules start by finding the boundary of the table caption. Using the previously found indicator as a starting seed, a cluster of lines is created using a distance threshold based on the average height of text lines. The first approach called *Cluster* is based on the characteristic table structure of rows and columns. Approaches based on similar observations can be found in literature [12, 15, 16]. Most scientific publications use either a one- or a two-column publication layout. This results in three different table positions: fully in left/right column or spanning both columns. The middle point of the caption bounding box is used to decide which of the three cases is applicable. This is based on the heuristic that the table is in the left column if the middle point is within the first 40% of the page width, and on the right if it is in the right 60%, and in the center otherwise. The percentage is calculated by dividing the horizontal location of the point by the page width. This also defines the horizontal dimension of the search space, because it is not necessary to consider the right column if the table is only in the left. Objects intersecting a straight line starting from the middle point will be considered. Limiting the vertical direction is achieved by another single-linkage clustering to the left and right, using the caption middle as starting point. Assuming the placement of tables and captions is consistent throughout the document, the final direction is defined by the direction that appears most often within the whole document. The second implementation is based on another frequent feature of tables, which was also exploited in [13]: *Lines*. Tables in scientific publications are rarely fully bordered, but horizontal lines between box labels and entries are common. Initially, the Hough Transform [19] is employed to detect straight lines on the page. Using the caption middle as a starting point, two straight-lined searches are conducted, one to the top and one to the bottom of the page. If a previously found line is encountered, two additional searches are started to the left and right at the height of the intersection point. All horizontal lines found at this height ($\pm$ 5 px) and longer than 150 pixels are sorted into a single "bin". If two lines have a gap of less than five pixels, they are regarded as being the same line. Encountering another caption ends the search for the reason

explained previously. Currently, each table caption has up to two sets of bins that could represent a table. Assuming at least one table has just one possible bin, all other bins can be associated per elimination. In case this assumption does not apply, the set containing more bins is seen as table. After this association process, all horizontal lines in one set are used to establish an initial bounding box. Within this boundary, the same process is executed based on vertical lines. These are employed to modify the current table boundary in vertical direction.

4.4 Recognition

The table model is based on the model defined in [8]. A graph has been chosen as the underlying data structure, which is advantageous for subsequent KIE. This is inspired by [1], where table elements are also represented as nodes within a graph and a graph neural network is employed to create the edge structure. The goal of this module is constructing a valid table model by converting textual elements into table *cells* and using a rule-based algorithm to develop the physical, functional, and semantic model of the table. This process is illustrated in Fig. 2. There is no clear distinction between physical and functional recognition in this approach. The algorithm is divided into three parts, namely aggregation of cells, recognition of columns and recognition of rows.

4.4.1 Cell aggregation

Singular `String` elements within the XML file are aggregated as table cells using the average distance between words and the average line height as heuristics. Additionally, each symbol sequence is analyzed whether it is a sub- or superscript. In this case, it is automatically assigned to the nearest cell. Afterwards, each cell is represented as a node within a directed graph. Using these cells, an initial row structure can be created based on a window of twice the height of the average text line.

4.4.2 Column recognition

The recognition of columns starts with recognizing column labels. To this end, horizontal lines that span the length of the bounding box are detected using the Hough Transform [2, 19]. The topmost line having at least one textual element above it serves as an indicator for labels. If no suitable line could be found, the elements of the topmost row are heuristically seen as column labels. If more than one layer (or row) of column labels was detected, it is necessary to create the structure between these elements. Starting at the lowest column label row, the distance to each predecessor in the row above is calculated for each element. The distance and the corresponding number of vertical lines crossed are used to calculate a penalty score for each pair. If the

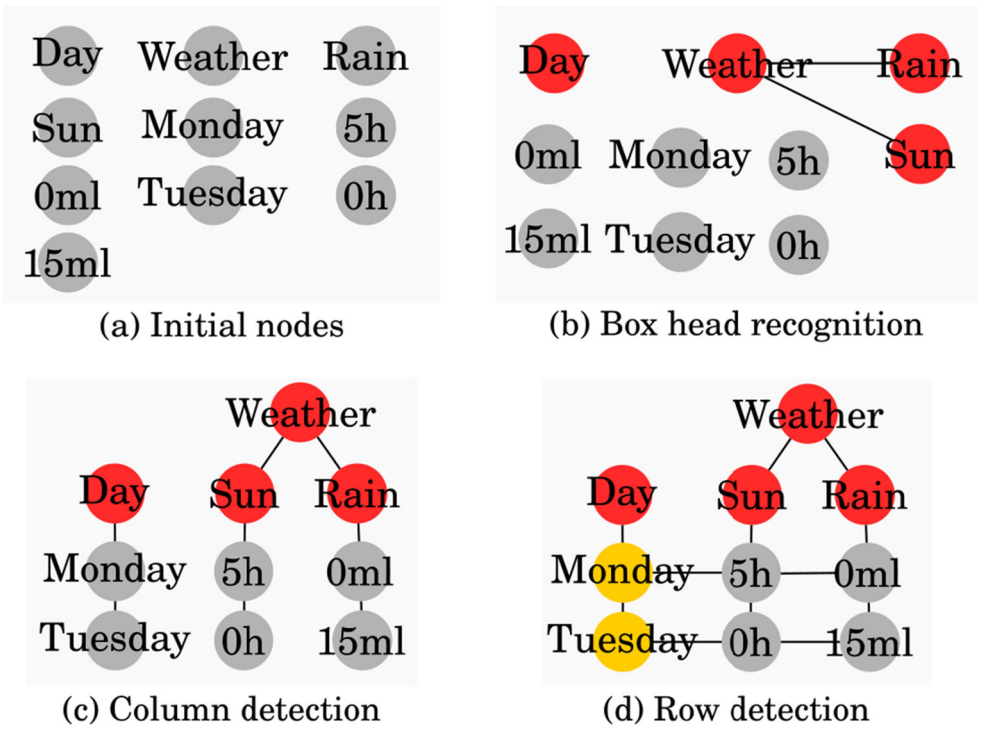

Fig. 2 Exemplary process of developing a table model using example Table 1. Each element is represented by a node within a graph. Using the previously described heuristics, column labels are recognized and the structure of the whole box head is developed. Nodes of the lowest level of the box head structure are employed to define the columns of the table. Rows are primarily recognized using a simple match-up algorithm that associates nodes of same-length columns. Irregular tables are handled using rules that transform them into a structure where the match-up algorithm is applicable

minimum of these scores is lower than the sum of a single line crossing penalty and three times the average distance between words, an edge is created. This creates a tree-like structure of column labels. Each leaf of this structure is defined as starting point of a column. All remaining cells are put into a column using horizontal overlap with these leaves. If a cell could not be associated with a column, a new column with a dummy column label is created and associated with this cell. Finally, all columns are sorted according to the vertical position of their elements, and column edges are created between neighboring nodes from top to bottom. Up to this point, both the physical and the functional model with regards to columns have been completed

4.4.3 Row recognition

The core of this step is a simple algorithm. Given a table with same-length columns and without any irregularities, the row structure can be completed by creating an edge between each cell and its right-hand neighbor in the following column. The cells of the first column can then be defined as row labels. However, this is not always the case, but every table can be transformed in such a way so that the algorithm is applicable to a part of it. While creating the dataset, three categories of irregularities have been observed, one having three sub-forms. Figure 3 contains examples for each category. These three cases are

Fig. 3 Examples of possible column structure irregularities. Three categories have been determined while creating the dataset, whereas one category has three possible forms. Two of these forms are nearly identical, because a semantic column as an in-between row is a variation of the case above. They are united by the fact that more cells usually imply multiple semantic columns in a single physical column. Fewer cells in the first column indicates that this column modifies the following column. All tables not covered by these two categories are put into the third category

	Category	Example Table
1	Irregular last column	Model / Score / Diff Baseline 70 A1 75 +5
2	First column with multiple semantic columns	Model / Score1 / Score2 Baseline B1 75 80 Architecture1 A1 76 81
	Semantic column as in-between rows	Model / Score1 / Score2 Baseline B1 75 80 Architecture1 A1 76 81
	First column with modifiers	Category / Model / Score1 / Score2 Baseline: B1 75 80; B2 75 80
3	Other	Model / L1 / Score2 Baseline 10 88 A1 10 90 Average 89

recognized and handled by rules. The recognition can be achieved by simply counting the number of cells per column and comparing all columns against these cases.

The first case handles tables with irregularities within the last column. All other columns are regular by definition and can be structured using the simple algorithm. Using vertical overlap to cells in previous columns, the remaining edges to cells of the last column are formed accordingly. The cells in the first column are defined as row labels. Columns with more or less nodes than the average nodes in the first column as well as "in-between" rows form the second category. More nodes than average within a column indicate that a single physical column contains more than one semantic column. It is assumed that these semantically different cells ("modifier") are represented by being in an extra row without other elements in the previously established initial row structure. An additional assumption is that modifiers modify all following rows until another modifier is found. Based on these assumptions, modifiers can easily be identified by iterating through the rows and columns. They are moved to an extra column and labeled as row labels and all edges between modifiers and modified nodes/cells are created accordingly. The nodes being modified are also labeled as row labels. If the first column contains fewer nodes than the average, the procedure used to construct the box head structure is applied here. Therefore, the associations between modifiers and nodes being modified are found using the distance between them. Both are again tagged as row labels. The overall third case encompasses all tables that do not fit into the other two categories. Here, the structure is recovered by just using vertical overlap as an indicator for elements being in the same row. Again, all cells in the first column are defined as row labels. Note that the simple algorithm as well as two of the cases define only a single layer of row labels by default. Here, some kind visual indicator (indentation or separate rows) is needed to recognize multiple layers of row labels. Determining a higher number of row labels usually requires a semantic interpretation and can therefore not be handled by a mostly abstract algorithm. Obviously the described transformation rules are incomplete, because the number of possible layouts is too high. Nonetheless, we expect satisfactorily results, because the layout of scientific articles usually follow standards, which can be exploited by the rules we created.

4.5 Ontology definition

Designing an ontology is an intricate and non-trivial process, because the more general the task, the more complex the ontology. Furthermore, disagreement may occur because the definition of semantics is in parts individual. However, as this contribution has a clearly defined goal as well as domain, it is possible to design an ontology that keeps the mentioned problems at a minimum. The most controversial topic is most likely the difference between the terms *Architecture* and *Variant*. These terms try to capture the origin of experimental results, whereby a *Variant* is a more concrete occurrence of an *Architecture*. For example, a result was produced by *Architecture* "BERT" of *Variant* "BERT-large". However, the boundaries are hard to define, because there are multiple possibilities how "BERT-large+xyz" could be handled in this example. Because tables are seen as singular objects, the same method is applied to resolve this problem. Initially, every "result producer" is *Architecture* and its own *Variant*. If two *Architecture*s share a common substring, the substring is set as *Architecture* and receives the *Variant*s. If an occurrence starts with a "+" or "-", it is defined as a *Variant* of the most recent *Architecture*. *Result*s in leaderboard creation [5, 9] usually are defined as a tuple (Task, Dataset, Metric, Score). This publication broadens this definition. However, because only the task of coreference resolution is considered here, the *Task* part of the tuple is omitted. Additionally to the existing *Metric* the *Measure* is modeled. In contrast to the common definition, *Architecture* and *Variant* is vital information here, because there is more than just a single result per paper. The final *Result* tuple therefore has the form (Architecture, Variant, Dataset, Metric, Measure, Score). An overview of the ontology, including the relationships between all classes can be seen in Fig. 4.

Using the table in Fig. 5 as an example, the following illustrates the use of each class. There is one *Architecture* called "BERT-large + c2f-coref" with two *Variants* "(independent)" and "(overlap)". The *Result* having the *Value* "77.5" also has the *Measure* "P" and the *Metric* "B^3" and was achieved on the *Dataset* "OntoNotes". The tuple containing every piece of information would therefore be ("BERT-large + c2f-coref", "(overlap)", "OntoNotes", "B^3" "F1", "77.5").

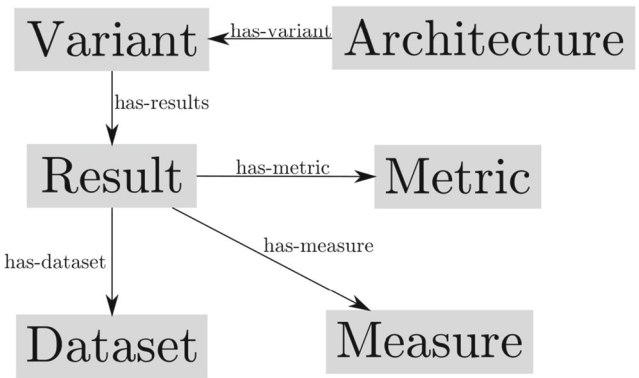

Fig. 4 Overview of the ontology. For reasons of clarity, individuals have not been included within this graphic

	B^3		
	P	R	F1
BERT-large + c2f-coref (independent)	76.5	74.0	75.3
BERT-large + c2f-coref (overlap)	77.5	70.9	74.1

Fig. 5 "Table 1: OntoNotes: BERT [. . .]" Part of Table 1 from Joshi, M, et al (2019) BERT for coreference resolution: baselines and analysis https://arxiv.org/abs/1908.09091 serving as an example for applying the ontology. The previously given caption is the start of the original table caption

4.6 Key-insight extraction

The generated models are filtered according to their relevance for the extraction by counting the occurrences of ontology keywords and ignoring every table with fewer than four occurrences. The remaining table graphs are processed by classifying each node according to the content. The spelling variants of individuals defined in the ontology and the cell content are used to determine which of the nodes within stub and box head are associated with which individual. Several rules define the difference between a *Variant* and an *Architecture*. For example, if there are two layers of row labels, it is automatically assumed that the first layer consists of *Architecture*s and the second of *Variant*s. Excluded are some defined keywords like "Baselines" or "Model Variants" that provide an umbrella term for a group of systems. Another rule is: systems starting with symbols like "+" or "-" are always *Variant*s. All nodes that did not match with any individual are seen as results. The process of constructing these objects consists of following along row and column edges and collecting all relevant individuals. If a property of a result could not be defined, the individuals having been found in the caption using the same keyword matching process are used to try finding a match. The result of this process is a data structure containing information about *Architecture*s, their *Variant*s and the associated *Result*s, described by *Metric*, *Measure*, *Value* and *Dataset*. Although values can be given in many different formats, the most common one is a percentage and it is assumed that all values are some form of percentage. The conversion between different formats ("50%" and "0.5") is done automatically, if a comparison between values is necessary. Because the ontology is based on OwlReady2 [10], the datastructure can be queried using SparQL and can therefore also answer the leaderboard question as described in [5, 9], although without naming the *Task*, which is implicitly added in our case.

5 Dataset

Finding a single dataset that covers all evaluation scenarios is difficult, because this contribution tries to evaluate as many steps as possible. Although there are a few TE datasets concerning scientific publications, the degree of annotation varies greatly and even the highest degree does not annotate information regarding the functional structure of the table. Although related, the dataset of [5, 9] and similarly TDMSci [6] can not be used for evaluating the KIE process within this publication. These are used to evaluate leaderboard creation, meaning a table annotation consists of the highest scoring results usually modeled as a tuple (task, dataset, metric score). All other results are not considered, resulting in most knowledge contained within the table being not accessible. Therefore, leaderboard datasets are neither suitable to evaluate the extraction of tables nor to evaluate KIE from tables.

Thus, creating a dataset that covers all evaluation scenarios was necessary.

A challenge of the creation process is generating a dataset that is a valid sample representing the real world as close as possible. In order to guarantee the validness, about 750 tables of scientific publications have been reviewed and categorized according to two criteria. The first criteria is complexity, roughly measuring the "hardness" of this table. It is based on the common cell representation as a list of triples $(columnlabels, rowlabels, value)$. Flattening this to $(coll_1, \ldots, coll_n, rowl_1, \ldots, rowl_m, value)$ yields a tuple whose order defines the complexity of the cell, whereby higher means more complex. A table is assumed to be within a certain complexity class T, if at least one cell tuple of the corresponding complexity was found. The second criteria is based on the physical layout of the table. There are many different ways how data can be represented in tables and some are more difficult for automated process, especially if there are many irregularities present. Determining the distribution of these layouts prevents the creation of a bias containing only complete and symmetric tables. The review yielded six common layouts of tables, the most common being "homogeneous" which includes all symmetric tables without any irregularities. These layouts disregard complexity, meaning that for example "homogeneous" (L0) includes tables having a single column label layer as well as multiple. Arguably, layout category L0 could have been split into one two separate categories, one encompassing tables with a single row and column

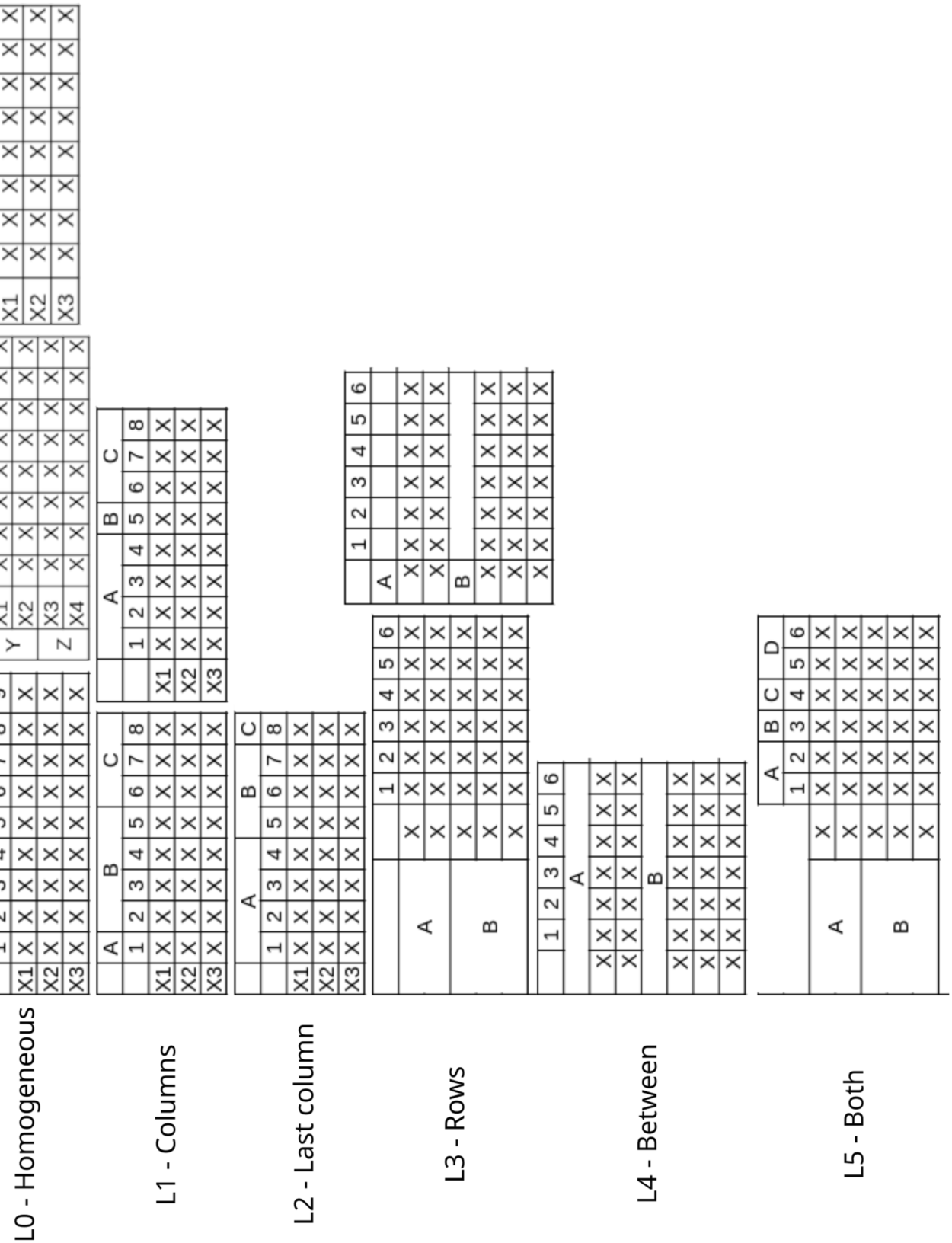

Fig. 6 Examples of each layout type. Homogeneous tables L0 are symmetric regarding their row and column arrangement. Column irregularities are found in L1 and L2, while L2 includes only cases where the last column differs from all other columns. L3 includes all row irregularities, with the exception of dividing rows, which are covered in L4. L5 describes tables with both row and column irregularities

Table 2 The percentage point difference between the distribution of complexity and layout categories between this dataset and the results of the review

	Category				
Layout	T2	T3	T4	T5	T6
L0	−1	0	−1	0	0
L1	0	0	−3	+1	0
L2	0	0	+3	0	0
L3	0	−1	+3	0	0
L4	0	0	−2	−1	0
L5	0	0	−1	+2	0

Positive values signal a surplus in this dataset, negative values signal the reverse. The difference range is ±3 percentage points

label layer and the other encompassing all other cases. However, it is much easier to recognize their structure, because of their symmetry. Consequently, we decided to create a group capturing all symmetric tables, regardless of their number of label layers. Examples of each group are illustrated in Fig. 6. The occurrences of complexity and layout categories and the difference between the distribution of this dataset and the results of the review are shown in Tables 2 and 3. The figure shows that the selection of the dataset is representative compared to the larger review set and the difference range lies between ±3 percentage points. Based on the acquired data, a dataset containing 208 tables from 49 documents has been created. Each dataset item contains metadata like the document id, page number, table number and the manually determined bounding boxes of the table as well as the caption. This metadata enables the evaluation of the detection and localization steps. The table is represented as a markdown-like string as well as a list of cell tuples in form of $(rowlabels, columnlabels, entry)$. The former is the evaluation basis for the physical model recognition, while the latter is used for the functional and semantic recognition. Furthermore, the caption is included as a string. In order to evaluate the KIE step, 31 of these 208 tables containing results on test sets have been annotated with a list of triples $(subject, predicate, object)$, so that all instances and relationships defined in the ontology are represented. There are several reasons why only 31 tables have been annotated. This annotation step depends on the employed ontology and cannot be easily transferred to other domains. Furthermore, the annotation process is very time consuming, because even small tables generate a significant amount of triples. The most important reasons is that the model of all tables processed by the KIE module is structured the same way. Consequently, the applicability of this KIE step can be inductively inferred. All steps have been manually executed or verified.

Note that the review and this dataset exclude textual tables, meaning tables containing a significant amount of text, whose primary purpose is not the presentation of numerical data.

Table 3 Number of occurrences of every complexity and layout category

	Complexity					
Layout	T2	T3	T4	T5	T6	Σ
L0 - Homogeneous	1	105	26	7	1	141
L1 - Column1	0	0	3	3	0	6
L2 - Column2	0	0	20	2	0	22
L3 - Row1	0	3	14	2	2	20
L4 - Row2	0	0	2	5	0	7
L5 - Both	0	0	0	11	1	12
Σ	1	108	65	30	4	208

Tables are sorted into a complexity category if at least one tuple of that complexity exists. Examples of each layout category are presented in Fig. 6

6 Experiments

This pipeline was designed with productive and mostly autonomous usage in mind. Consequently, the output quality of the system can not only be judged by raw numbers as it is also important to know the amount of work required to correct the output. The error rate, meaning the ratio of erroneous tables and the overall number of tables, is a representation of the amount of work needed. To this end, the measure *Table Error Rate* (TER) has been introduced to compare the number of erroneous tables against the overall number of found tables. An erroneous table describes a predicted table that produced false positives and/or false negatives. Tables without errors are tables that only produced true positives.

6.1 Detection

The mainly autonomous usage that this pipeline was designed for requires that table indicators are found reliably. Even if the boundaries of a table could not be established successfully, knowledge of its existence allows the extraction of the single corresponding page, obviating the need to look through the whole document manually. The detection rate is evaluated using the number of tables per page, counting excess tables as false positives, missing tables as false negatives. Therefore, precision and recall signal a general tendency to predict too many or too few tables. As other systems do not include this evaluation measure, the number of objects per page resulting from a full run of the system has been counted (Table 4).

Open-source solutions only find about a third of all tables, if they are applied to pages without any additional information. Like previously assumed, using the heuristic of a common string signaling the start of table captions seems to be a good indicator for the presence of tables. However, the performance of this approach depends on the table caption layout, because it has to match one of the defined templates. An extension of the system is easily achieved by adding new layouts to the engine if encountered. The applicability of this approach can also be seen in Section 6.4.

6.2 Recognition

The resulting model of the recognition module allows the individual evaluation of several of its aspects. Although two localization approaches have been developed, in order to guarantee a fair comparison between all systems and to avoid redundancy, the experiments have been conducted with the assumption of pre-established bounding boxes.

6.2.1 Physical evaluation

The physical structure is evaluated using unique adjacency relations between neighboring nodes/cells, commonly called proto-links (PL) [3, 7]. Assuming a fully bordered table, neighboring cells are cells that share a border and form a single proto-link, meaning PLs (A, B) and (B, A) are seen as being identical. Precision and recall can thus be defined as $\frac{\#correctPL}{\#detectedPL}$ and $\frac{\#correctPL}{\#totalPL}$.

Camelot, TableSeer and Tabula have been evaluated using this metric in addition to our approach. It is assumed that a perfect bounding box is given in order to guarantee a fair comparison between all systems, although TableSeer can not be fed with additional information and is therefore evaluated on its raw output.

The most important goal of the table extraction step is getting a correct semantic model of a table. Because our model transforms irregular tables into regular tables, our physical representation differs from the actual one, although there is no semantic difference. Therefore, physical errors are disregarded if the resulting semantic model contains no errors.

The results in Table 5 show the performance of our approach being higher than all other approaches. Additionally, our error rate is half of the best performing comparison model.

Table 4 Evaluation result of the detection step

				# detected tables	
	Precision	Recall	F1	Gold	Overall
Camelot	0.27	0.86	0.41	179	667
TableSeer	0.88	0.72	0.79	150	170
Tabula	0.39	0.75	0.51	155	399
KIETA	1.00	1.00	1.00	208	208

Precision and recall are calculated based on the number of predicted and ground-truth tables per page

Table 5 Evaluation result of the recognition step, evaluated regarding the physical, functional and semantic aspect of a table

	Measure	**KIETA**	Camelot	TableSeer	Tabula
Physical	Precision	0.97	0.97	0.67	0.82
	Recall	0.94	0.89	0.54	0.57
	F1	0.95	0.93	0.60	0.67
	TER	52/208	114/208	135/153	206/208
Functional Row	Precision	0.93	-	0.40	-
	Recall	0.92	-	0.35	-
	F1	0.92	-	0.37	-
	TER	33/208	-	136/153	-
Functional Column	Precision	0.89	-	0.33	-
	Recall	0.93	-	0.24	-
	F1	0.91	-	0.28	-
	TER	30/208	-	136/153	-
Semantic (inferred physical and functional model)	Precision	0.82	-	-	-
	Recall	0.81	-	-	-
	F1	0.81	-	-	-
	TER	62/208	-	-	-
Semantic (perfect physical and functional model)	Precision	0.91	-	-	-
	Recall	0.90	-	-	-
	F1	0.90	-	-	-
	TER	41/208	-	-	-

Camelot and Tabula only develop a physical representation of the table, while TableSeer also extracts column labels (row labels have been generated by generally defining cells of the first column as row labels). All steps are also evaluated regarding the table error rate (TER). Semantic evaluation has been conducted twice, once in a normal end-to-end setting and once assuming perfect results of previous steps

6.2.2 Functional evaluation

As mentioned before, labels define the semantic structure of tables and give context to entries, which is important regarding subsequent knowledge extraction. It is therefore beneficial to evaluate the recognition rate of these labels. To this end, the sets of predicted row and column labels are compared to the ground truth. The strings are compared directly, so any match is counted as true positive, and mismatch as false positive or false negative respectively. The resulting precision and recall give information about whether too many or too few labels are recognized.

TableSeer was the only external approach attempting to create the functional model of the table. However, only column labels were classified, so each entry of the first column is defined as row label to create results regarding row label classification.

The results in Table 5 illustrate that our column label recognition performs slightly worse than the recognition of row labels. A common error of column label recognition is merged cells, while an additional problem of row label recognition is the definition of surplus layers of row labels. The performance of TableSeer shows a clear difference to the heuristics of our method. Both row and column label recognition scores being similar and manual review of some predictions indicates that its heuristic for column label detection is also simply taking the first row. Obviously, these heuristics are not enough to process more sophisticated tables.

6.2.3 Semantic evaluation

The semantic model is the basis of key-insight extraction and therefore the most important of all models for the purposes of our contribution. The evaluation is based on the representation of cells as tuples, including all associated row and column labels and the value $(row_1, \ldots, row_n, col_1, \ldots, col_m, entry)$. Such a tuple is created by simply following all row and column edges in reverse direction starting at an entry and aggregating all labels along the way. True positive are only full matches, partial matches are false positives. As previously mentioned, the stub head is ignored. No other system could be used for comparison, as no system could be found that created a semantic table model.

The last part of Table 5 includes the results of this experiment. Evidently, the performance is slightly lower than the performance of previous experiments. This follows

Model	Score
Baselines	
B1	80
B2	81
A1	90
Other	
T1	84

Fig. 7 Example of a table that cannot be recognized correctly because of the assumptions that have been made. "Baselines" would modify "A1" even though it should only modify "B1" and "B2"

the process of the pipeline, because semantic recognition builds on top of everything else. As a consequence, previous steps define the upper boundary of what can be achieved here.

However, some tables cannot be fully recognized even the physical as well as functional recognition delivered perfect results. The assumptions made while recognizing a table were described previously. One of these is concerns modifiers, meaning cells that semantically belong to an extra column but are merged with usually the first column. All nodes not identified as modifier are associated to the most recent modifier, which can cause errors. This is best seen in an example. Given the table in Fig. 7, the modifier "Baselines" would be associated to all nodes until another modifier ("Other") is found, resulting in an relationship between "Baselines" and "A1". In order to avoid these errors, the heuristics for recognizing modifying nodes have to be improved.

6.2.4 Error discussion

Tables 6 and 7 are overviews of the error distribution regarding complexity and layout categories. Generally, the more unusual the table (complexity or layout), the higher is the error rate. The error rates of the simplest and most common forms T3 and L0 are 17% ($\frac{18}{108}$) and 23% ($\frac{32}{141}$) regarding semantic evaluation. However, $T4$ and $L2$ are neither the most complex nor the most complicated layout, but their error rate are high in comparison.

A common error influencing both physical and functional recognition concerns elements that are wrongly aggregated to cells. This includes incorrect merges as well as separations. For example if the distances between textual elements within the box head are to small, they are merged into a single cell, causing errors in both recognition parts. Additionally, there are several instances of table cells with line breaks. In these instances, the cell content does not fit within the cell width and spills into the following line, falsely creating a new row with a new "cell". Solving this problem without any semantic information is almost impossible, because these instances are indistinguishable from other layouts (e.g. Fig. 6, L3/2).

Incorrect relationships between cells can also be caused by large distances between cells. Incorrect means both a non-existing relationship where there should have been one and genuine erroneous relationships. Both physical and functional recognition are impacted by this problem. Similarly, small distances can cause identical problems.

The recognition of column labels depends on the heuristic that there is a horizontal line dividing these cells from normal data entries. If no line is found, the topmost row is assumed to describe the column labels. Because

Table 6 This table categorizes the errors in Table 5 (rows "TER") with regards to the complexity of tables

Step	Data	T2	T3	T4	T5	T6	Overall
Physical	Absolute	0	13	25	10	4	52
	TER	0%	12%	38%	33%	100%	25%
Func. Row	Absolute	0	14	9	6	4	33
	TER	0%	13%	14%	20%	100%	16%
Func. Col.	Absolute	0	11	12	4	3	30
	TER	0%	10%	18%	13%	75%	14%
Sem. (inferred)	Absolute	0	18	30	10	4	62
	TER	0%	17%	46%	33%	100%	30%

The absolute number of errors and the Table Error Rate (TER) in percentages are given for each step. The last column describes the total error and TER of a step and equals the TER presented in Table 5. The TER is calculated using the dataset distribution in Table 3

Table 7 This table categorizes the errors in Table 5 (rows "TER") based on the table layout

Step	Data	L0	L1	L2	L3	L4	L5	Overall
Physical	Absolute	26	2	9	8	2	5	52
	TER	18%	33%	41%	40%	29%	42%	25%
Func. Row	Absolute	20	0	4	5	1	3	33
	TER	14%	0%	18%	25%	14%	25%	16%
Func. Col.	Absolute	19	1	5	3	1	1	30
	TER	13%	17%	23%	15%	14%	8%	14%
Sem. (inferred)	Absolute	32	2	10	11	2	5	62
	TER	23%	33%	45%	55%	29%	42%	30%

The absolute number of errors and the Table Error Rate (TER) in percentages are given for each step. The last column describes the total error and TER of a step and equals the TER presented in Table 5. The TER is calculated using the dataset distribution in Table 3

the line recognition process is a probabilistic method, an incorrect horizontal line is possibly defined as the separating line. This being an early step within the recognition process, the whole subsequent process is disrupted.

Errors concerning row labels were mostly the result of previous errors. The main area of improvement is the aggregation of textual elements to cells, being both the first step of the whole recognition process and the main cause of the described errors.

6.3 KIE evaluation

The ontology used to define the desired knowledge is modeled using OWL and RDF. Accordingly, the evaluation is modeled using RDF-like triples (subject, verb, object). Each of the three aspects as well as the full triple is evaluated individually. For example, given the list of predicted triples as well as the ground truth, it is assumed that the "subject" of the RDF-like triple should be evaluated. To this end, every predicted subject is compared to the remaining ground truth. If the subject could be matched, it is counted as true positive. All ground truths without a match are counted as false negatives and vice versa regarding remaining predicted tuples.

The experiment is conducted twice, once in an end-to-end scenario (*E2E*) and again using the ground truth data of the previous step as input (*Gold*). Because the ground truth contains knowledge gaps as some tables are not completely self-descriptive, the *Gold* scenario is evaluated using another set of ground truth data without knowledge gaps that cannot be closed using only tables and their captions.

Table 8 shows that both systems produced output on 30 of the 31 tables, meaning that one table could not be processed by the module. Predicates could be extracted without many problems, followed by subjects and objects. This is expected behavior, because "object" has the highest amount of variation, while "predicate" has the lowest. The part of the triple performing the worst (in this case "object") sets the upper bound for the full triple evaluation.

While recall is an important statistic, precision is more relevant in this scenario, because it is better that a piece of knowledge is not extracted at all than it being extracted erroneously and consequently corrupting the knowledge base. Assuming this point of view, the TER can additionally be evaluated considering only tables containing false positive errors, reducing it by 7 and 10 percentage points, respectively.

The ground truth data being used for this evaluation contains only knowledge that is actually included in the tables, meaning that for example the dataset is not annotated if it is not contained within either table body or caption. "Full" is the evaluation with ground truth data containing knowledge *not* mentioned within the table or caption. The comparison between both sets allows for measuring the impact of tables that do not follow the desirable practice of being self-descriptive. The highest difference is two percentage points considering the recall of full triples.

The small difference between full triple and the worst of the other three evaluations suggests that the errors are spread across a small number of triples.

Furthermore, the difference between gold input extraction and extraction on the output of the systems suggests that the problem lies either in the chosen method or the tables themselves.

The error analysis shows three categories: ambiguity, incorrect resolutions and miscellaneous errors. Considering the "E2E" scenario, there are of course propagated errors caused by the table extraction system. Our system currently does not have the ability to resolve ambiguity. For example, our ontology contains the term "CoNLL" as dataset as well as metric, if it is used in combination with "F1". However, this can currently only be detected if both terms are within a single cell. Otherwise, they are separately detected.

The differentiation between *Architecture*s and *Variant*s is the leading cause for resolution errors. However, the

overall impact of this particular error is small, because the relationship between *Result*s and *Variant*s are still valid. More relevant is that the information contained within the caption is not correctly resolved. For example, if the system detects an ontology instance within the caption, it happens that this instance is associated to columns that actually describe another ontological instance.

6.4 Field experiment

To demonstrate the practical use of our system we conducted a field experiment on real data that has not been pre-processed. To this end, the resulting 184 PDF documents of the query "Coreference Resolution" on arxiv.org have been processed by the pipeline. 94 of these documents contained at least one table that could be processed with the ontology defined within this contribution. The 184 documents contained 947 tables overall of which 142 tables contain processable knowledge concerning the results of experiments. All other tables contained other kinds of knowledge. The overall statistics can also be seen in Table 9. Each row shows the number of tables that should have been processed at this step and the actual number of tables that produced any output. Clearly the mechanism deciding whether a table is important or not is not reliable as twice as many tables than desired have been deemed relevant.

All PDF documents were processed by the pipeline and the resulting knowledge data structure was queried using SPARQL. This is not a precise analysis of the pipeline but rather a demonstration of a possible workflow as well as the applicability and advantages of the tool. To this end, the main question posed in the introduction "What system is currently SOTA on any dataset?" has been formulated as a SPARQL query, which can be seen in Fig. 8. The result on the right shows the top ten results, naming the arXiv ID, the *Variant* name as well as the average F1 score and the

Table 8 Evaluation of the key-insight extraction step using precision (P), recall (R) and F1 based on semantic triples (subject, predicate, object)

	Subject			Predicate			Object			Triple			TER	
	P	R	F1	P	R	F1	P	R	F1	P	R	F1	(n)	(p)
E2E	0.955	0.836	0.892	0.979	0.858	0.914	0.947	0.829	0.884	0.933	0.817	0.871	$\frac{19}{30}$	$\frac{16}{30}$
Gold	0.969	0.869	0.916	0.993	0.891	0.939	0.956	0.857	0.904	0.944	0.846	0.892	$\frac{18}{30}$	$\frac{16}{30}$
Full	0.969	0.852	0.907	0.993	0.873	0.929	0.953	0.837	0.891	0.940	0.826	0.880	$\frac{18}{30}$	$\frac{16}{30}$

Every part of the triple as well as the full triple is compared to the ground truth. The (n)ormal table error rate (TER) is the ratio of tables containing errors and all processed tables, while (p) signifies that only tables with false positive errors are counted. "E2E" represents the extraction of key-insights based on the output of the TE pipeline, while "Gold" uses perfect table extraction results. "Full" can be compared to "Gold", but is evaluated using additional ground truth data that can currently not be extracted

Table 9 Number of processed tables per step, whereby processed means the production of any output

	Optimal	Processed	Percentage
Detection	947	942	0.995
Localization	947	906	0.957
Recognition	947	876	0.925
Key-insight	142	306	2.155
Deemed not relevant	805	521	0.647

Of 947 tables overall, 942 have been found and 876 could be produced by the pipeline. However, two times more tables than desired have been sent through the key-insight extraction module, suggesting that the employed heuristic based on the number of instance/keyword appearances has to be adjusted

```
SELECT DISTINCT ?ID
  ?x
  ?data
  ?value
WHERE {
  ?x a Variant .
  ?y a Result .
  ?x has_results ?y .
  ?y has_metric Average .
  ?y has_measure F1 .
  ?y has_value ?value .
  ?data a Dataset .
  ?y has_dataset ?data .
  ?super has_variant ?x .
  ?super used_in ?ID .
}
ORDER BY DESC(?value)
```

```
 ,ID,variant,dataset,avg_f1
0,[1],CorefQA,CoNLL2012,83.4
1,[2],CorefQA(Wuetal.,2020),
    ↪ OntoNotes,83.1
2,[1],CorefQA+SpanBERT-large,
    ↪ CoNLL2012,83.1
3,[1],--SquadPre-training,CoNLL2012
    ↪ ,83.1
4,[1],--QuorefPre-training,
    ↪ CoNLL2012,82.7
5,[1],CorefQA+SpanBERT-base,
    ↪ CoNLL2012,79.9
6,[3],SpanBERT,CoNLL2012,79.7
7,[4],Simplifiede2e-coref(Ours),
    ↪ OntoNotes,79.7
8,[2],Baseline(Joshietal.,2020),
    ↪ OntoNotes,79.6
9,[1],c2f-coref+SpanBERT-large(
    ↪ Joshietal.,2019a),CoNLL2012
    ↪ ,79.6
10,[1],--SpanBERT,CoNLL2012,79.6
```

Fig. 8 Left: Query for best average score on any dataset; Right: Result. The references of mentioned papers are given below

1. Wu, W et al (2020) Coreference Resolution as Query-based Span Prediction. https://arxiv.org/abs/1911.01746
2. Xia, P et al (2020) Incremental Neural Coreference Resolution in Constant Memory. https://arxiv.org/abs/2005.00128
3. Xu, L and Choi, J D (2020) Revealing the Myth of Higher-Order Inference in Coreference Resolution. https://arxiv.org/abs/2009.12013
4. Lai, T M et al (2021) End-to-end Neural Coreference Resolution Revisited: A Simple yet Effective Baseline. https://arxiv.org/abs/2107.01700

dataset it was achieved on. The results of the system seem to be consistent and plausible to the best of our knowledge, although some tables presenting results on development sets have been processed. This shows that the mechanism deciding the relevancy of a table has to be improved to only process desired tables. However, as mentioned already no detailed analysis has been conducted.

7 Conclusion

Our publication has presented a number of important findings regarding the extraction of key-insights from tables. The performance of previously available TE systems is not good enough for the purposes of extracting knowledge. Consequently the development of a specialized system that can handle the oftentimes complex layout of this kind of tables is required.

To this end a table extraction pipeline[6] has been developed, having the goal of extracting tables as well as their captions from digital PDF files concerning coreference resolution. Furthermore, a graph-based table model is created that captures the physical and especially the semantic structure of the table. Using this model, knowledge defined by an ontology is extracted as a searchable datastructure.

A dataset[6] with 208 tables from 49 documents has been created in order to evaluate every step of the pipeline as well as the key-insight extraction.

Experiments conducted on that dataset show that a rule and heuristic based TE pipeline can produce results surpassing currently available open-source projects in the domain of sciences. Furthermore, it is possible to use the created table model graph for the extraction of high-quality key-insights defined by an ontology.

The focus for future work lies on improving the recognition module in particular, because it is the foundation upon which the knowledge extraction is built, but also the hardest problem of the whole extraction pipeline. Possible extensions include for example handling more cases of row/column irregularities and improving employed heuristics in terms of performance and robustness. Furthermore, reducing false positives and therefore increasing the precision of the key-insight extraction module has high priority, starting with the development of an ambiguity resolution method.

[6] https://gitlab2.informatik.uni-wuerzburg.de/kieta/kieta

As has been shown, some tables are not self-explanatory and therefore do not contain every piece of information necessary to understand their content. However, analyzing a table within the context of its whole publication, the text as well as other tables, is necessary to gain access to the full knowledge. The current system is not able to handle these cases as tables together with their captions are considered self-contained objects. It is a long term goal to extend the system to consider the text of the publication itself to find knowledge that could not be extracted from the table. A common example of missing knowledge is the dataset the experiment was conducted on, which is sometimes only mentioned in the text body of the publication.
A short term goal is implementing an interactive component, where users can add additional data and correct errors, so that they have a practical useful tool for remaining up-to-date in their research field concerning the increasing number of relevant publications with evaluations.

Funding Open Access funding enabled and organized by Projekt DEAL.

Declarations

Competing of Interests Author Frank Puppe is a member of the editorial board of Applied Intelligence (APIN).

References

1. Chi Z, Huang H, Xu HD et al (2019) Complicated table structure recognition. preprint at arXiv:1908.04729
2. Duda RO, Hart PE (1972) Use of the hough transformation to detect lines and curves in pictures. Commun ACM 15:11–15. https://doi.org/10.1145/361237.361242
3. Göbel M, Hassan T, Oro E, Orsi G (2012) A methodology for evaluating algorithms for table understanding in PDF documents. In: DocEng. ACM Press, pp 45–48. https://doi.org/10.1145/2361354.2361365
4. Grobid (2008)
5. Hou Y, Jochim C, Gleize M, Bonin F, Ganguly D (2019) Identification of tasks, datasets, evaluation metrics, and numeric scores for scientific leaderboards construction. In: ACL. Association for Computational Linguistics, pp 5203–5213. https://doi.org/10.18653/v1/p19-1513
6. Hou Y, Jochim C, Gleize M, Bonin F, Ganguly D (2021) TDMSci: A specialized corpus for scientific literature entity tagging of tasks datasets and metrics. In: EACL. Association for computational linguistics, pp 707–714. https://doi.org/10.18653/v1/2021.eacl-main.59
7. Hurst M (2003) A constraint-based approach to table structure derivation. In: ICDAR. IEEE Comput. Soc, pp 911–915. https://doi.org/10. 1109/icdar.2003.1227792
8. Hurst MF (2000) The interpretation of tables in texts. PhD, University of Edingburgh
9. Kardas M, Czapla P, Stenetorp P et al (2020) AxCell: Automatic extraction of results from machine learning papers. In: EMNLP. Association for computational linguistics, pp 8580–8594. https://doi.org/10.18653/ v1/2020.emnlp-main.692
10. Lamy JB (2017) Owlready: Ontology-oriented programming in python with automatic classification and high level constructs for biomedical ontologies. St Heal T 80:11–28. https://doi.org/10.1016/j.artmed. 2017.07.002
11. Li M, Cui L, Huang S et al (2020) TableBank: Table benchmark for image-based table detection and recognition. In: LREC. European language resources association, pp 1918–1925
12. Liu Y, Bai K, Mitra P, Giles CL (2007) TableSeer Automatic table metadata extraction and searching in digital libraries. In: JCDL. ACM Press, pp 91–100. https://doi.org/10.1145/1255175.1255193
13. Nurminen A (2013) Algorithmic extraction of data in tables in PDF documents. Master, Tampere University
14. Oelen A, Stocker M, Auer S (2020) Creating a scholarly knowledge graph from survey article tables. In: Ishita E, Pang NLS, Zhou L (eds) ICADL. Springer International Publishing, pp 373–389. https://doi.org/10. 1007/978-3-030-64452-9_35
15. Perez-Arriaga MO, Estrada T, Abad-Mota S (2017) Table interpretation and extraction of semantic relationships to synthesize digital documents. In: DATA. SCITEPRESS - Science and technology publications, pp 223–232. https://doi.org/10.5220/0006436902230232
16. Rastan R, Paik HY, Shepherd J (2019) TEXUS: A unified framework for extracting and understanding tables in PDF documents. Inform Process Manag 56:895–918. https://doi.org/10.1016/j.ipm.2019.01.008
17. Ren S, He K, Girshick R, Sun J (2017) Faster r-CNN: Towards real-time object detection with region proposal networks. IEEE Trans Pattern Anal Mach Intell 39:1137–1149. https://doi.org/10.1109/tpami.2016. 2577031
18. Singh M, Sarkar R, Goyal P, Mukherjee A, Chakrabarti S (2018) Ranking state-of-the-art papers via incomplete tournaments induced by citations from performance tables. preprint at arXiv:1802.04538
19. Sinha P (1962) Recognizing complex patterns. Nat Neurosci 5:1093–1097. https://doi.org/10.1038/nn949
20. Xie S, Girshick R, Dollar P, Tu Z, He K (2017) Aggregated residual transformations for deep neural networks. In: CVPR. IEEE, pp 5987–5995. https://doi.org/10.1109/cvpr.2017.634