TUM Data Innovation Lab

Munich Data Science Institute

Technical University of Munich

&

Steering Lab by Horváth & Partners GmbH

Final report of project:

Natural language processing for intelligent data mining and augmentation of probabilistic graphical networks

Authors	Esmée Oosterlaar, Niklas Lüdtke, Vaibhav Jain and Simon Lohrmann
Mentor(s)	M.Sc. Olena Schüssler, M.Sc. Marie Matthäus and Dr. Raphael Kozlovsky
Project Lead	Dr. Ricardo Acevedo Cabra (MDSI)
Supervisor	Prof. Dr. Massimo Fornasier (MDSI)

Aug 2022

Abstract

The interest in creating knowledge graphs of entity relations has been steadily growing since 2012 [Inc04], due to the many applications in research and businesses. They can be especially useful during early phases of research or monitoring a market or competitor for trends over time.
To create such knowledge graphs, an end-to-end pipeline that starts with PDF documents as input and ends with a complete knowledge graph as output is crucial. These pipelines however, are complex systems consisting of many individual models. In this project we are exploring each part of the pipeline to convert a document into a knowledge graph, finding best models for the problem definition and fitting them into one pipeline with a special focus on analyzing environmental research papers.

Our pipeline consists of five main parts, where each part was first developed independently and then everything was merged in one code base. The first part is abstract extraction, where the summary of the paper is extracted in order to retain the most important information, while making the next steps faster and easier to train, due to the reduced size of data. Next, coreference resolution is used in order to capture more relations spanning over multiple sentences and later enabling combinations of relationships in the knowledge graph. For example, an abbreviation of an entity, like *UBA* for *Umweltbundesamt* should refer to the same entity, therefore we replace the abbreviation with the actual name in the text. Afterwards the entities are extracted with a combination of a self-developed rule-based approach, mainly focusing on subjects and objects with their respective modifiers, and a pre-trained transformer-based model from the spaCy library. For each pair of entities in a sentence an unsupervised model called SelfORE classifies their relation into a cluster. Even though the model is unsupervised, i.e. no labels are needed during training and consequently the output only produces cluster ID's for each sample, we apply two different techniques to recover labels for the relationships from the training data. These techniques use the most frequent n-grams between entities in a cluster and the root words in these sentences, respectively. The results from the entity and relationship extraction are visualized in a knowledge graph by the last module of the pipeline.

This pipeline is an easy to use prototype showing the great potential of these end-to-end frameworks. The modularity of the pipeline also enables a quick deployment of different models in the respective pipeline module, i.e. focusing on multilingual support for relationship classification or a deep learning approach to abstract generation.

Contents

1 Introduction

Natural Language Processing (NLP) is a branch of Artificial Intelligence that focuses on the ability of computers to understand spoken and written human language [Lut21].
It is widely used in everyday life, for example in spam detection, auto correction, chatbots, translation, speech recognition and so on. Its applications are available in almost every language and in this day and age Natural Language Processing is essential.
NLP has existed for over decades, starting in the late 1940s, and has been tremendously growing in recent years. It started out with complex sets of hand-written rules that were replaced in the late 1980s by statistical models [Lou20].
Human language is full of ambiguities like homonyms, sarcasm, metaphors, sayings etc. The complexity lies in understanding its full meaning, with human's intent and sentiment. Even for humans it is hard to get an accurate understanding of these anomalies when learning a new language, let alone for a software written by programmers [IBM20].
Our project focuses on a certain branch of NLP called knowledge graphs (KG). In the case of this project, these graphs visualize the relations between entities in a text, making them understandable for humans at a glance. They make information more accessible for the user without the need of reading an entire document. They can either give an idea of the contents of a single document or be used to analyze a number of documents about a certain topic. For example, one could filter out only entities that are useful for one's research and their connections to other entities. Looking for frequent relations then gives a first idea of how the entities interact and which ones are important. Another important benefit of a knowledge graph is that it makes text documents understandable to a computer, making them more accessible for automatic analysis and processing. This opens the door to access a huge amount of data, only available in text form, benefiting corporations in gathering information and analyzing situations more efficiently and more precisely.
The objective of this project is to investigate relationships between entities extracted from scientific papers of the German Environment Agency. This is done by building an NLP model that consists of several steps. To visualize these relationships, a knowledge graph is built, where the nodes represent the entities and the edges the relationships between them. The aim is to have the most important information of the text documents at a glance.

2 Pipeline and Data

2.1 Outline

Due to the nature of the problem it was possible to structure the project as a pipeline. An overview of the pipeline can be seen in Figure 1. The first chapters explain the theoretical background of each part of the pipeline, which is followed in Chapter 6 by a detailed explanation of how these parts were assembled into a model. The current chapter describes the input and justifies why abstract extraction is used. The main components of the pipeline are discussed in Chapter 3, which consists of coreference resolution and entity extraction, and in Chapter 4, where the relationship extraction module is explained. What a knowledge graph consists of is then explained in Chapter 5.

Figure 1: Diagram of the pipeline

2.2 Data

The text documents used as input are scientific papers in PDF format from the German Environment Agency (Umweltbundesamt), which are written in English as well as in German language. Their lengths vary from three to 40 pages.
The four papers used as examples in this report are scientific opinion papers called "The Revision of the REACH Authorisation and Restriction System" [Ros+22], "The Zero Pollution Action Plan as a chance for a cross-regulatory approach to pollution prevention and reduction" [Con+21], "Obsolescence - Political strategies for improved durability of products" [Age17] and "Auswirkungen des Klimawandels auf die Verbreitung Krankheitserreger bertragender Tiere (exotische Stechmcken)" [TLJ20].

2.3 Abstract Extraction

Since the summary or abstract of a paper contains its most important information, a knowledge graph of only these paragraphs will suffice to get a good overview of the paper, while keeping the graph clear. The input of the model can in theory be any PDF. However, due to the predefined structure of scientific papers, which usually have a summary or abstract in the beginning pages, fine-tuning a model to these properties will give more comprehensive results. I.e. a PDF with a clearly labelled abstract works best, due to the rule-based approach needed here.
The heuristics applied to PDFs are limited to very little high-level information, due to the formatting of PDFs. The available information are the text, font size, formatting such as bold and italic and position information for each letter. As will be shown in Chapter 6, even with this little information, one can build a model that consistently finds the abstract in a scientific paper.

3 Entity Extraction

To be able to extract relationships, it is first necessary to find entities where such a relation is likely to occur. In linguistics, entities are real world objects such as people, places etc with a proper name. Looking at the relation between any two entities in a sentence is computationally infeasible, especially since many entities consist of more than a single word.

3.1 Coreference Resolution

The first step of entity extraction is coreference resolution, an NLP technique needed when two or more expressions refer to the same entity. For example, in the sentence "The German Environment Agency promises it will proceed with its plans soon.", "The German Environment Agency" and "it" refer to the same underlying entity. Coreference resolution finds all the entities with the same meaning, clusters them together and replaces each with their reference entities. By performing coreference resolution long-range dependencies in the text can be handled and any duplicate edges in the knowledge graph are avoided. For example in Figure 2 an edge between "The German Environment Agency" and "Zero Pollution Action Plan" will represent the same relationship as an edge between "UBA" and "Zero Pollution Action Plan" and therefore "UBA" is a redundant node in the knowledge graph and should be replaced by "The German Environment Agency".

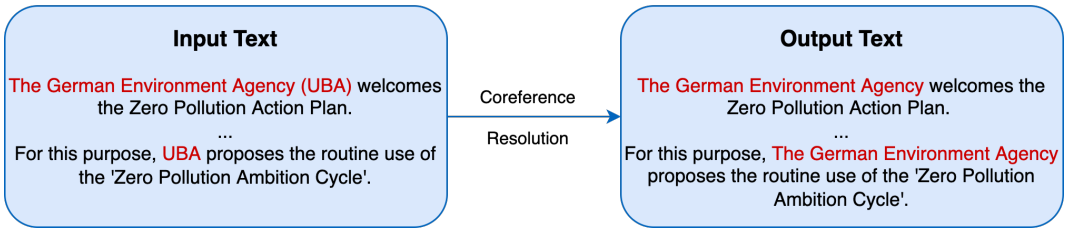


Figure 2: Coreference resolution example

3.2 Named-Entity-Recognition (NER)

The next step is to actually extract the relevant entities. Most of the relationships of interest are between entities that correspond to organizations, companies, countries or similar, so-called named entities. There are many open source models available trying to extract named entities from a text. These models are called named entity recognition (NER) models.
Many NER models are similar to an iterated dilated convolutional neural network structure. Here, a small stack of dilated convolutional neural networks is applied recurrently to broaden the context of a token and prevent overfitting [Str+17].
In recent years, BERT-based NERs have been developed, that tend to have a better performance. BERT [Dev+18] is a transformer-based model that is trained on a huge text corpus using masked language modeling combined with next sentence prediction. It achieves state-of-the-art performance on many NLP tasks by simply fine-tuning the pre-trained model for the downstream tasks [Mat19].

3.3 Rule-Based Approach

Another approach to extract entities is rule-based entity extraction, which is an NLP technique that uses predefined rules to identify entities that follow a certain pattern [Gom21]. These rule-based approaches only work on sentence level and not on paragraph or document level. For the extraction of single word entities, one could apply Part-of-speech-Tagging (POS).
POS tagging is a popular NLP process, that assigns a tag to each word in the text based on the type of word (part of speech). The tags that are assigned are based on the definition of the word and the context in which the word is used. Extracting entities that consist of multiple words is more complicated, so in this case sentence dependency parsing is used, since Part-of-speech-Tagging alone would not suffice.
Dependency parsing is the process of analyzing the grammatical structure of a sentence and finding the type of dependencies (relationships) between different words in the sentence. The result of dependency parsing is a dependency tree where on the nodes we have head words and words, that modify those heads. The relationship between them acts as the edge attribute.
More details on how to extract entities using dependency trees are explained in Chapter 6.

4 Relationship Extraction

The goal of relationship extraction is to classify the relation of entities in a sentence, such as

Entities:	The German Environment Agency, The Zero Pollution Action Plan
Relationship:	welcomes

With the entities extracted in Chapter 3, the next step is to assign a relation to every pair of entities in a sentence. There are different ways of modelling this problem.

Classification model Simply put, the input is a sentence with two marked entities, and the goal is to classify their relation into one of the predefined relation classes. Training data is available with different numbers of relation classes. For example, the T-REx dataset [Els+18] has 642 different relations, where the TypeRE dataset [Fer20] only includes 28 different relations.

Clustering model Instead of using predefined relation classes, the sentences can also be clustered into ones with similar properties. For example, it could be useful to not distinguish between "*brother of*" and "*sister of*", but denote this relation as "*sibling of*". Clustering methods could be tuned, such that these properties are fulfilled. Since there are no relation labels available, one has to revert to other techniques to create labels from the training data.

In the following chapter different approaches to the relationship extraction problem are described.

4.1 Rule-based Model

The sentence to which the entities belong usually contains some information of the relationship. A simple approach is therefore to try to extract this information from the words in the sentence. One can define fitting heuristics, based on POS tagging and dependency trees described in Chapter 3.3. These heuristics have the advantage of being easy to understand and explainable. However, generalizing them to a large text corpus is a difficult task and requires a lot of hand-crafting of rules.

4.2 Transformer Model

Transformer models are nowadays widely available through packages like Huggingface [Wol+19]. Using a simple classification layer on top of the hidden state outputs of the model and training it on labelled data has been proven to work well for relationship extraction, e.g. in [Yan+21], where the model achieved an F1-score of 0.8959 on a clinical dataset. There are also more sophisticated architectures, which are designed for specific sub-problems, like document level relationship extraction, e.g. in [XCZ21].

4.3 Self-supervised Model: SelfORE

Unsupervised or more specifically self-supervised models have the benefit of being more flexible in terms of input data and in this case also the output dimension. Since the domain of scientific papers is not well-studied in the area of NLP and relation extraction, this approach would benefit from more data that is specific to the problem and also generalizes easily to other applications, i.e. internal documents of companies, since unlabelled training data is sufficient.
This specific self-supervised model uses soft-assignment based on KMeans clusters to create pseudo-labels in order to train a classification feed-forward neural network. The complete model, called SelfORE, consists of three modules and was proposed in [Hu+20]. As seen in Figure 3 the transformer model (here: BERT [Dev+18]) creates hidden states used in adaptive clustering and the classification module. Then, the adaptive clustering algorithm creates pseudo-labels, that are used in the training loop of the classification module to train it and the transformer via backpropagation. Naturally, the focus here is the adaptive clustering module, due to its novelty.

Adaptive clustering The goal of adaptive clustering is to assign each sample, with their respective hidden state $H = \{h_1, h_2, \cdots, h_N\}$, to a cluster label. This label will eventually be used in the training of the classification module. However, in traditional clustering techniques, the number of clusters have to be predefined. In the setting of this problem the number of clusters is not always given, especially in the case of unlabelled training data. Therefore, only semantically meaningful clusters should be assigned and the model has to be insensitive to the initialized number of clusters, i.e. the output dimension.
Adaptive clustering consists of an encoder network, which projects the hidden states $h_i \in \mathbb{R}^{h_R}$ to a latent representation $z \in \mathbb{R}^{h_{AC}}$ and soft-assignment to K cluster centroids learned by KMeans. In detail, KMeans is first applied to the latent representation to find K initial centroids $\{\mu_k \in \mathbb{R}^{h_{AC}}\}_{k=1}^{K}$. Afterwards, a similarity measure based on the Student's t-distribution is calculated for the latent representations $\{z_n\}_{n=1}^{N}$ and the centroids. The loss function $\mathcal{L}_{AC}$ is the KL divergence between the similarity measure and an auxiliary distribution, which emphasizes high confidence assignments. The pseudo-labels are then assigned by finding the cluster with the highest probability for each sample.
Training is only performed for the encoder part of the network (the transformer parameters are not affected). One training loop of the full model starts by training the adaptive clustering module until the cluster assignments do not change significantly anymore. Afterwards, normal training of the transformer and classification module is performed for a set number of epochs.

Pre-Training of Encoder The encoder layers integrated in the adaptive clustering module are initialized with an auto-encoder, which is trained before the first iteration of the training loop for the whole model. This ensures that information is preserved after the dimensionality reduction.

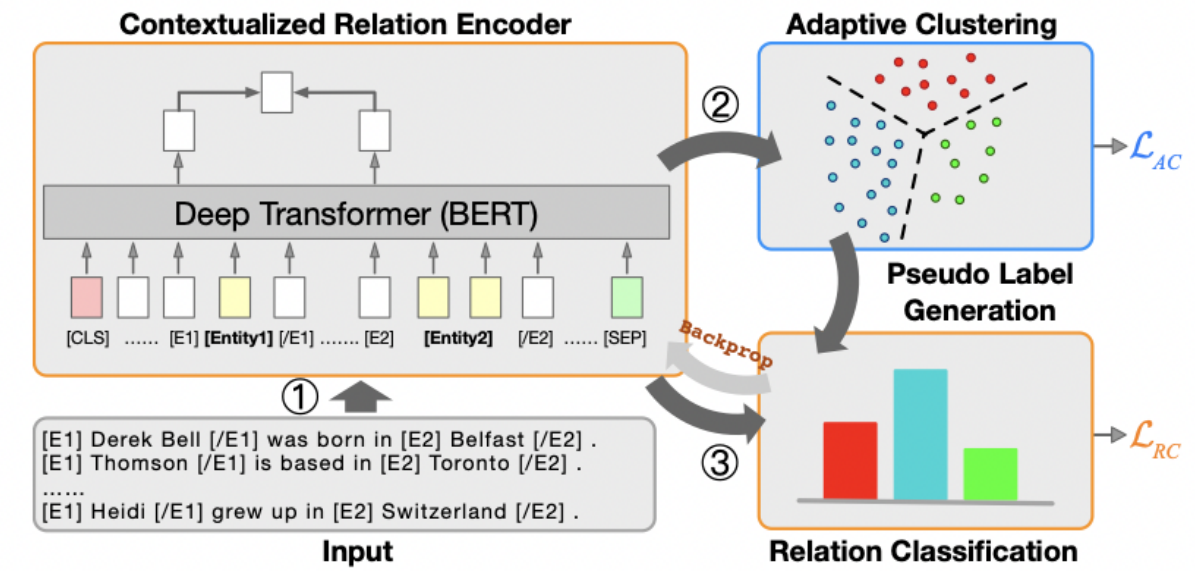


Figure 3: SelfORE structure [Hu+20]

Weak-Supervision In order to take advantage of available labelled training data, the model can be extended to be first trained in a supervised mode. This could be done in order to ensure that the transformer and classification module indeed learn important semantic relations between entities, rather than random ones. Therefore, we can substitute the KMeans algorithm in the adaptive clustering step with the true labels, encoded as one-hot vectors. During this "warm-up" step, the transformer parameters can be frozen even in the training of the classification module.

Inference In deployment, the classification module produces cluster labels for each sample by choosing the cluster with the highest probability $c_i = \text{argmax}_{k \in K} l_i$. If using weak-supervision, one could simply take the labels from the labelled training data. However, this mitigates the positive aspects of using an unsupervised model, that it is more flexible in terms of relations. Another approach is to use all samples classified into a cluster during training, to create a description of the relation. Using the text between the entities in each sentence, which most likely contains some description of the relation, the most frequent n-gram, can provide a reasonable description. Therefore, after the model is trained, the training data can be used to generate these labels for each relation cluster, by finding the most frequent n-gram in the text between entities for each cluster.
Experiments in [Hu+20] have shown that these n-grams can provide valuable information for the user.

4.4 Discussion

Choice of Model The relationship extraction module has to satisfy a wide range of characteristics, directly related to the problem statement:

- **Inference speed & simplicity** In general, the speed of inference should be reasonable, such that users do not have to wait for results for hours. In fast-paced work

schedules it is often important to have access to information as quickly as possible. Hence, a long inference time would be counterproductive to the goal of this project. The model could either run locally on a laptop or in the cloud on a GPU, the first being the simpler option.

- **Adaptation to new relations** The range of relations in environmental studies, but also in general, is vast and cannot be quantified exactly. Relations defined in some publicly available datasets, like TypeRE, might not cover all relations specific to the problem. Therefore, the relationship extraction module should be able to adapt or be adapted quickly to new relation classes in training data.

- **Business impact** The results of the model should focus on being useful in a business case, i.e. give valuable and unbiased information to the user. Therefore, the relations have to be as precise and understandable as possible to an untrained user.

These metrics will be useful to decide on which model fits the purpose of this project the best and how some possible downsides could be approached for the model to have more impact.

Choice of Transformer In most relationship extraction models discussed above, a transformer is used to extract information from the input text, therefore the choice of transformer has a huge impact on the final performance of the model. This also allows to optimize for different metrics:

- **Speed of inference** This metric could be especially important for deployment, if the model is run on laptops or even mobile hardware. Inference for big models could take from a few seconds to minutes. When analyzing multiple documents this could lead to inference times of hours, mitigating the advantages of such model in the workplace.

- **Language agnostic models** In the case of a multilingual work environment, which is the case in many international corporations, documents in different languages could be interesting to the user. Therefore, a multilingual model would give great improvements on the inference side and the implementation side. Another approach would be to first classify the language of a document, and then applying different models based on this prediction, one for each language. This needs more testing, training and research on the implementation side however, making the creation of such models more labour intensive.

- **Precision** If only the precision of the inference plays a role, one could in theory use the biggest language models. However, there is no research on which models perform the best in the different relationship extraction models discussed here. The default choice in most research papers is the BERT transformer model [Dev+18].

.

5 Knowledge Graph

A knowledge graph is a network that consists of two main components: nodes and edges. It models the underlying relations between the entities, where the nodes represent the entities and the edges represent the relationship between those entities.
The edges can be directed, which characterizes an asymmetric relationship between two entities, or undirected, which characterizes a symmetric relationship. Moreover, it is possible for a node to have multiple edges, and in case there exists no edge between two nodes, it means that there is no relationship between them. Hence, it only displays the interconnected entities. An example of a (part of a) directed knowledge graph can be found in Figure 4.
A knowledge graph can be heterogeneous or homogeneous. In a homogeneous graph, all entities and all relationships are of the same type, which is for example the case in a social network graph while in a heterogeneous graph, the entities and relationships can be of different types. [Kam20]
Knowledge graphs can be created from different types of text: from a paragraph, a whole text document or even from several text documents at once, etc. It all depends on the use case. One might be interested in comparing the different knowledge graphs of individual text documents, or perhaps in relationships between multiple text documents. Moreover, one could focus and filter on entities that are considered particularly important or that are of the same type, and there are many other possibilities.
Knowledge graphs come in different shapes and sizes. For example, the Google knowledge graph consists of millions of entries. [Inc21]
Since a knowledge graph visualizes the entity pairs and their relations, it makes it easy to get an overview of a text document, without having to read the whole document. It puts the data in a context that is interpretable and easily understandable and therefore makes it simple to share with others. Generating a similar output by hand would not be feasible, which is why it is of great advantage to create a knowledge graph using NLP techniques. These graphs are useful not only for humans, but also for computers, as they make the text documents understandable for them as well, which simplifies automatic analysis and processing.

Figure 4: Knowledge graph example

6 Prototype

In Chapters 2, 3 and 4 the theory behind the main parts of the end-to-end pipeline were explored. The main challenge consists of integrating these parts into one model, that consistently gives good visual results. Due to the large number of parts in the pipeline, each of them has to be carefully tuned, to give useful outputs for the next part. In this chapter this process is described in detail, giving an overview of the whole pipeline, discussing the training involved for different models and how the final graph is created. The results of the end-to-end pipeline are discussed at the end of the chapter.

6.1 Current Models

Abstract Extraction As discussed in Chapter 2.3, the difficulty of extracting the abstracts or summaries from the given text documents, lies in the fact that these are in PDF format. To tackle this problem, we use several steps of heuristics that we defined based on sample data.
In a first step, we count the number of letters for each font size in the document. With this distribution, the font size with the most letters, is defined as the font-size of the paragraphs. From this point, larger font sizes are defined as headings and smaller sizes as sub-paragraphs with the possibility to have multiple sizes in each category, e.g. *Heading 1*, *Heading 2*, etc. With this information we are then able to distinguish between the font size of the main text and the rest of the text, like titles and footnotes (see Figure 5). The next step is to search for key words, like "Abstract" or "Summary" for the English papers and "Abstrakt" or "Einleitung" for the ones in German language, in the headings that were extracted. If we search in the whole document there will be cases where these key words appear in paragraphs or table of contents.
The paragraphs following the header with a key word until the next heading is then taken as the summary. However, there are cases where the model only extracts part of the summary, therefore we set a minimum character count for the abstract. If this is not met at first, the next paragraph is also added to the output.

With this simple model, we are able to extract every abstract in the data available. In cases where there are no abstracts, a simple fallback rule is used, where either all paragraphs from the first page are used, or the whole document, if it has fewer than two pages.

Coreference Resolution The coreference resolution model is applied on the abstracts extracted from the scientific papers. We use AllenNLP's BERT-based large coreference predictor, which returns the text after replacing every entity with its referenced entity.

Entity Extraction In the entity extraction step, the processed string is split into sentences, since the models used here only work on a sentence level. We use the sentence tokenizer from the NLTK library. To each of the sentences, two different types of models are applied: a deep learning model and a rule-based model.
To extract the named entities, we use deep learning models from the spaCy library; one for the papers in English language and one for the papers in German language. We use

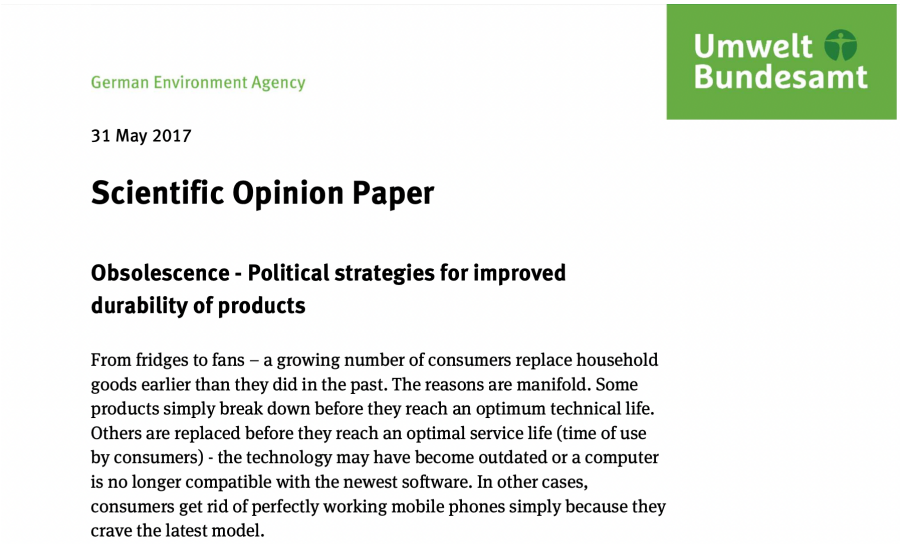


Figure 5: Part of the front page of a PDF. We are filtering headings by considering the most frequent font size

the spaCy transformer model to extract named entities in the English text, while in the German case we opted for a multilayer CNN model, since the German transformer model did not perform well. In addition to organizations, countries, and similar entities, spaCy NER models also extract dates and cardinal numbers, entities that are unlikely to have meaningful relationships and are therefore removed from our entity list. One example of the extracted entities is given in Figure 6. Here, one can see that the NER model extracts many important entities where a relationship is likely. In the very first line the entities "The European Green Deal's" and "the Chemicals Strategy for Sustainability" are being extracted which is very promising since the model is trained on web data and applied on scientific data.

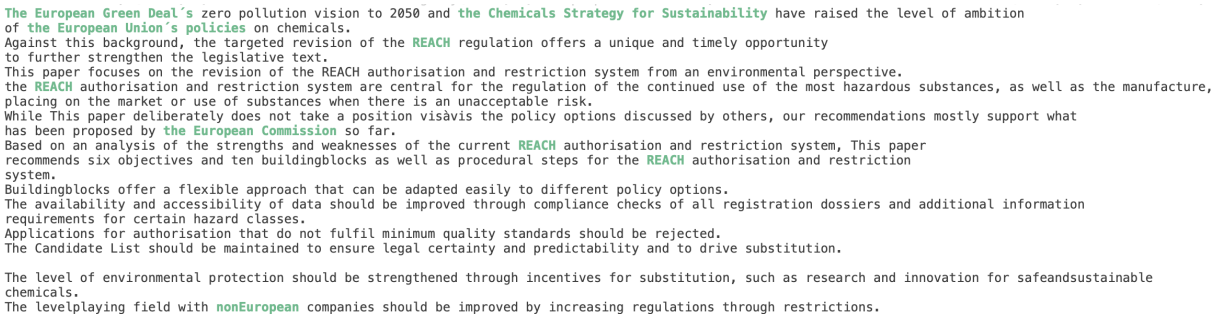


Figure 6: Named entities extracted by the spaCy transformer model

Since we would like to have as many relevant entities as possible and the deep learning approach only extracts the named entities, we decide to give a closer look at rule-based approaches for entity extraction. The first step is to produce dependency trees with the dependency parser from the spaCy library. After careful analysis of the dependency trees of various sentences, we formulate the following entity extraction rule:
"Extract the subject and all objects along with their modifiers, compound words and punctuation marks between them." [Jos19]

We implement this rule into Python by using conditional statements. For the German model we follow a similar approach, but since the grammar differs, we slightly change the rules of which entities to extract. Figure 7 shows entities extracted using rule-based approach.

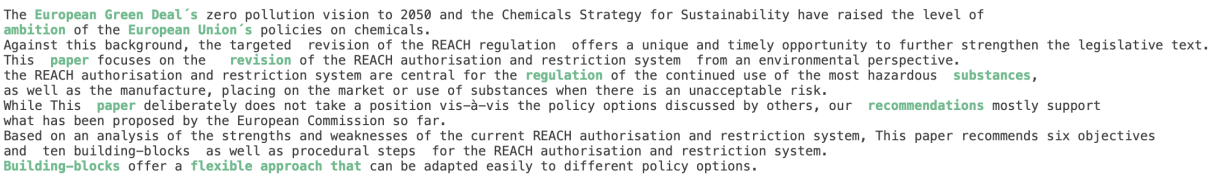
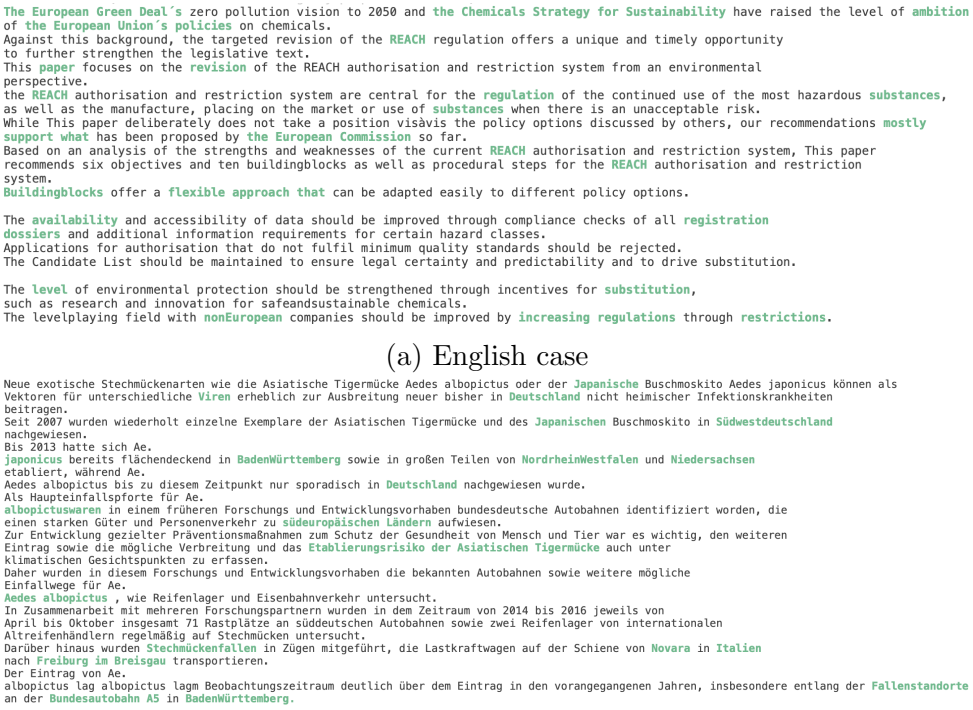


Figure 7: Entities extracted by rule-based approach

We observe that neither the deep learning model nor the rule-based approach were able to extract all the entities on their own. For this reason we decide to combine the list of entities extracted from both the models to generate the final entity list. However, it is not possible to blindly combine the two lists, as some entities may be repeated in both methods and need to be identified and included only once. Apart from handling matching entities, we must also handle entities that are almost similar. For example, "German Environment Agency" and "The German Environment Agency" should be considered as one and the same entity and not as two separate entities. To solve this problem, we develop a text similarity function using Sequence Matcher that gives the probability of how similar two texts are. Sequence Matcher counts the total number of n-grams that are similar in 2 strings by varying "n" from 0 to length of smaller string. It than return the ratio of number of similar n-grams to total possible n-grams between 2 strings. We discard any entity that has a similarity probability greater than 0.7 with any of the existing entities in the list. The results of combining the deep learning approach with our rule-based approach can be seen in Figure 8a. The rule-based approach added some more entities like "ambition" in the first line or "Buildingblocks" in the last line of the first paragraph. In Figure 8b a German example of the extracted entities is given. Here, our model also extracts important entities like "Aedes albopictus" which is a mosquito or "Etablierungsrisiko der Asiatischen Tigermcke" as well as German states and cities.
Since our model is looking for relations inside a sentence, we can omit sentences with less than two entities. For the remaining sentences we create a text file, where each line consists of a sentence with two marked entities. This is necessary since our model can only examine the relationship between two entities at a time. For example, if there are four entities in a sentence, the sentence will be repeated six times with two different entity pairs marked each time. A snippet of a text file can be found in Figure 9 below. In each line the considered entities are marked with start and end markers $[E1]...[/E1]$ and $[E2]...[/E2]$, respectively. As explained before, the first sentence is given six times as there are four entities: "The European Green Deal's", "the Chemicals Strategy for Sustainability", "ambition" and "the European Union's politics".

Relationship Extraction For relationship extraction we use the SelfORE model described in Chapter 4.3. It takes the text file created in the entity extraction module as input. The output of the model is the cluster assignment for each sentence in the input text file. The model performs best, when the number of relations is known beforehand,



(a) English case

(b) German case

Figure 8: Combination of NER with rule-based approach. Our model finds many entities which can further be used to extract important relationships

The [E1]European Green Deal's[/E1] zero pollution vision to 2050 and [E2]the Chemicals Strategy for Sustainability[/E2] have raised the level of ambition of the European Union's policies on chemicals.

The [E1]European Green Deal's[/E1] zero pollution vision to 2050 and the Chemicals Strategy for Sustainability have raised the level of [E2]ambition[/E2] of the European Union's policies on chemicals.

The European Green Deal's zero pollution vision to 2050 and [E1]the Chemicals Strategy for Sustainability[/E1] have raised the level of [E2]ambition[/E2] of the European Union's policies on chemicals.

Figure 9: Marked entities

however if this is not the case, one can set the number of clusters to an arbitrarily high number and due to the preference of high confidence cluster assignments, the samples will only be assigned to a few clusters.
Since the model only outputs cluster IDs for each sample, we decided to use N-grams to find the most likely relationship between each tuple inside a cluster. Experiments in [Hu+20] have shown that 3-grams provide the most valuable information for the user as shown in Figure 10, if the available data used to generate 3-grams is large enough.

Training of Models Training is needed for the classification layers of the SelfORE model, since there do not exist pre-trained layers and the transformer models from Huggingface come with a newly initialized classification layer on top of the transformer model. For the training we chose the T-REx dataset, since the available PDF documents did not contain enough sentences to train a model. Since the T-REx dataset consists of $\sim$ 6.2 mil

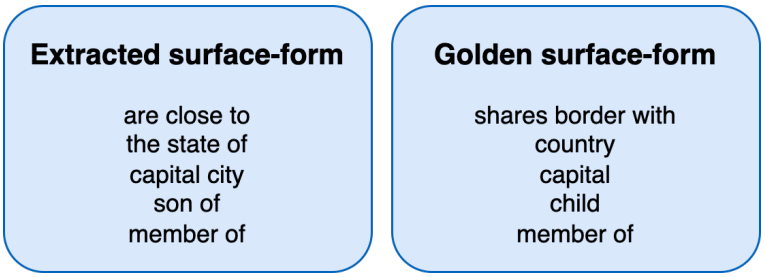


Figure 10: Results of 2 & 3-grams vs the ground truth relationship as shown in [Hu+20]

sentences, we randomly chose a small subset of 14662 sentences for training and testing.

Knowledge Graph Generation In our case, we consider a directed heterogeneous knowledge graph. The entities are obtained by the entity extraction while the relationship extraction acquires the associated relationships. These are then stored in a dataframe, such that the entity pairs and their relationships are in the same row. Next, we use the networkx library to generate a graph out of this. We generate two different graphs: one where the relationships are determined using the N-gram approach, and another where they are determined using the root word approach. We decide to not allow self-loops and remove these.

6.2 Alternative Models

During the course of the project we also investigated other models that can be used for the relation extraction module of the pipeline. In this section we discuss these additional models and when they could be useful in deployment.

Root-Based Approach After analyzing various sentences, we came to the conclusion that the main verb of the sentence is the most likely relationship between the entities if the sentence contains only two entities. However, if multiple entities are present in the sentence the main verb cannot define the relationship between all the entity pairs. To overcome this issue we extract the sub-sentence between each entity pair and define the verb of that sentence as the most likely relationship between the given entities. To find the verb of the sentence, we use spaCy's large BERT-based dependency parser and take the root word as the relationship type. The results of this approach are better than just keeping the main verb of the whole sentence as the relationship between all entity pairs. The result on a sample statement can be seen in Figure 11.

```
[E1]Building-blocks[/E1] offer a [E2]flexible approach that[/E2] can be adapted easily to different policy options.
Building-blocks offer a [E1]flexible approach that[/E1] can be adapted easily to different [E2]policy[/E2] options.
This [E1]paper[/E1] focuses on the [E2]revision[/E2] of the REACH authorisation and restriction system from an environmental perspective.
This paper focuses on the [E1]revision[/E1] of the [E2]REACH authorisation and restriction system[/E2] from an environmental perspective.
```

Figure 11: Root-based relationship example

The key advantage of this model over deep learning approaches is its explainability and that it doesn't need much data to produce results. We analyze its results in Chapter 6.3. However, the scope of this model is limited, since it is not able to improve its performance on large datasets and there are no deep connections and structures learned.

Supervised Model for Relation Classification We looked at various supervised models for relation classification and trained our own model by following the ideas from [Xu+15]. The main idea is to utilise the shortest dependency path (SDP) which is the shortest path from one node to another node in the dependency tree of a sentence. The motivation of using the SDP between entities is based on the idea that it usually contains necessary information to identify their relationship. The structure of our model is shown in Figure 12.

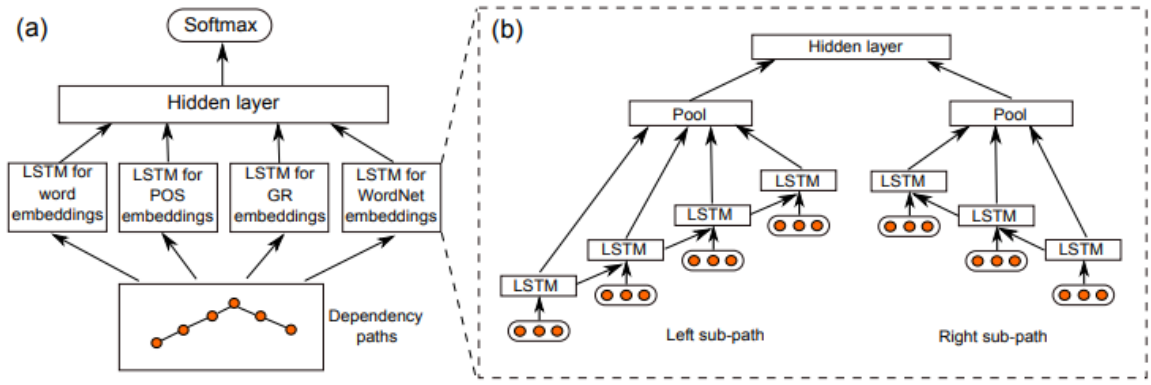


Figure 12: Supervised model [Xu+15]

We use 4 different type of embeddings as an input to out classification model :

- Word embeddings taken from a GloVe model.
- Embeddings for hypernyms from Wordnet of each entity which are again taken from GloVe pretrained embeddings.
- Embeddings for POS tags which are generated using LSTM units.
- LSTM embeddings for dependency relations in the SDP.

The inputs to the LSTM units are one-hot encoded vectors of POS tags/relation tags for each word in the SDP. We concatenate all the embeddings and feed them to a k-class classification model where k depends on the type of dataset we are using for training. We train the model end to end i.e. both the LSTM units and the classification units are trained together using the same forward pass, backpropagation cycle.
In our project we lacked enough labeled data that is specific to the domain of environmental research papers, therefore this approach could not be applied properly. However, if these requirements are satisfied, this model will most likely achieve higher accuracy than the self-supervised model used in the pipeline.

6.3 Results

For every PDF file a separate knowledge graph is generated. In Figure 13 the knowledge graph of the paper "The Revision of the REACH Authorisation and Restriction System" is depicted. The model extracts some one-on-one relationships as well as larger clusters. The

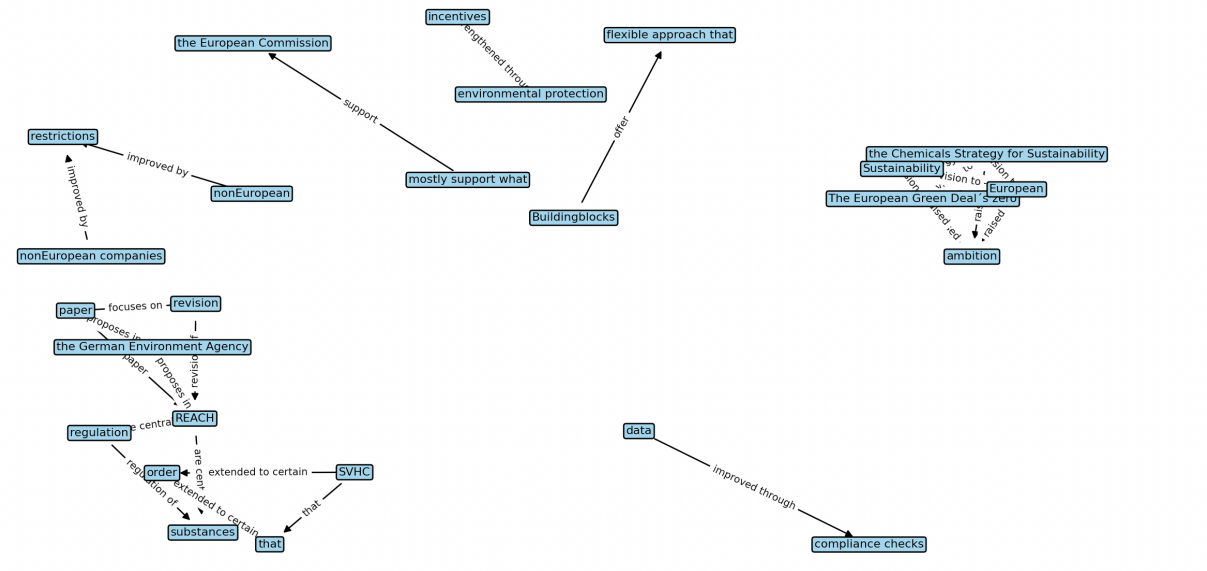


Figure 13: Root word knowledge graph

entities and their relationships seem promising. We take a closer look at three examples in the graph.

The first example can be seen in Figure 14. The corresponding sentence to this relationship is the following: "The availability and accessibility of data should be improved through compliance checks of all registration dossiers and additional information requirements for certain hazard classes.". We see that the intention of the sentence is very well captured. "data" represents "The availability and accessibility of data", "improved through" represents "should be improved through" and "compliance checks" represents the rest of the sentence. Our model takes the most important information and puts them in the graph. The entities and their relationship do not consist of too many words, which is desirable, since it makes the graph clear and comprehensible.

In Figure 15 another example is shown. This part of the graph represents the sentence "Building-blocks offer a flexible approach that can be adapted easily to different policy options.". The extracted relationship between "Buildingblocks" and "flexible approach that" is "offer", which is again an adequate representation of the sentence. The hyphen in "Building-blocks" gets removed by our entity extraction step and both entities originate from the rule-based approach. This example shows that there is still room for improvement on the rule-based entity extraction, because a better entity would be "flexible approach"

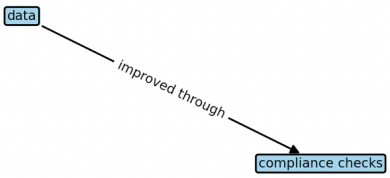


Figure 14: The relationship 'data improved through compliance checks' is a positive example of the root based knowledge graph

without the word "that".

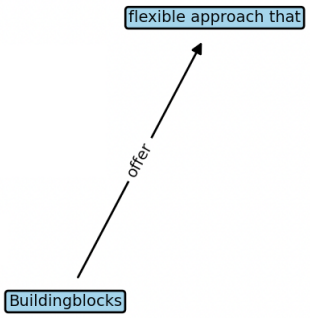


Figure 15: The relationship 'offer' is a positive example that we are able to extract semantics between entities

The last example is a cluster with several entities and relationships. It contains entities and relationships that span over two different sentences in two different paragraphs of the summary, which was accomplished by our coreference resolution: "This paper focuses on the revision of the REACH authorisation and restriction system from an environmental perspective. [...] In conclusion, the German Environment Agency proposes in this paper a set of ambitious revisions to the REACH authorisation and restriction system, ...". The entity pair and their relationship [paper, focuses on, revision] are very meaningful and we even get a relationship explaining what actually gets revised, namely "REACH". Then in the second sentence "the German Environment Agency proposes in this paper" also gets represented well in the graph by [the German Environment Agency, proposes in, paper]. Moreover, [the German Environment Agency, proposes in, REACH] also somehow makes sense. We see once again, that our model does fairly well in taking the most important information out of the sentences and visualizing this in a graph. Again, some improvements can be made, especially since [paper, paper, REACH] does not make that much sense.

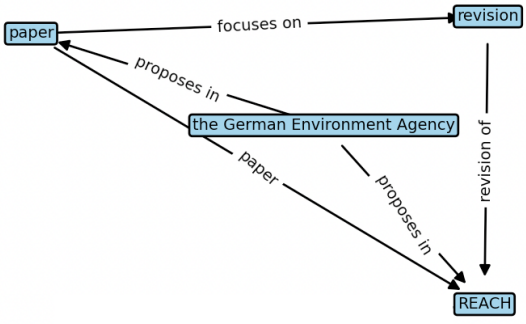


Figure 16: A larger cluster with positive examples such as 'paper focuses on revision of REACH'

The corresponding 3-gram knowledge graph can be seen in Figure 17. The entities and the number of relationships stay the same. All but two clusters are assigned to the 3-Gram

"0", consisting of the words "the","Chemicals" and "Strategy". The other relationship "1" consists of "nonEuropean", "companies" and "should", so both relationships do not seem to represent a fitting relation type. This is likely due to missing training data and one should expect better results when a larger corpus is used for generating labels, as shown in [Hu+20].

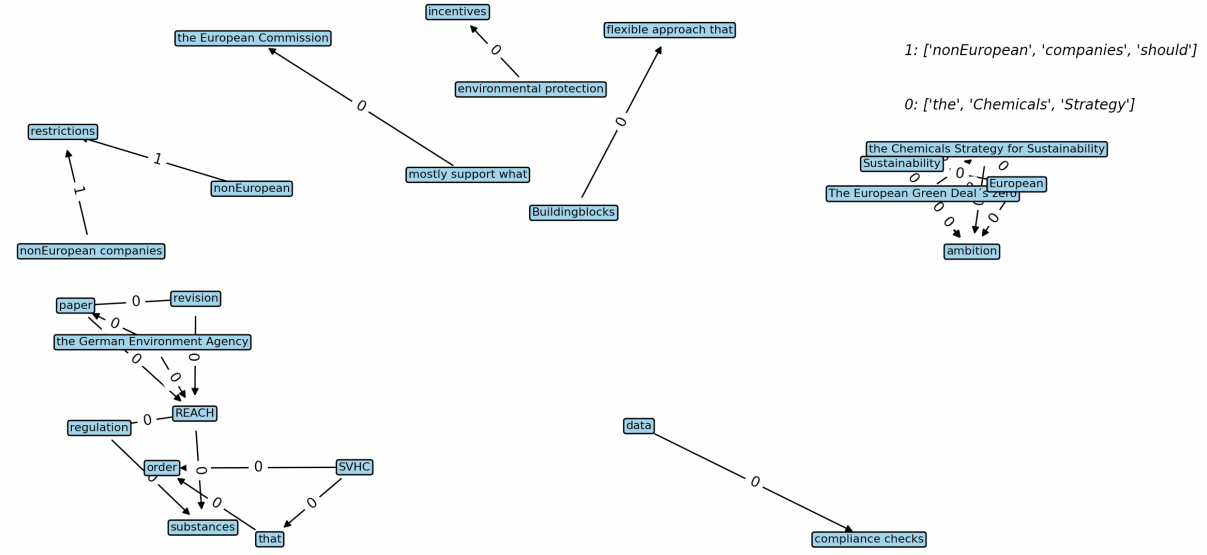


Figure 17: 3-Gram knowledge graph

Taking a look at a graph of a German text document in Figure 18, we can see that some relations between entities are extracted decently despite the model being trained on an English dataset. For example, there are relations between 'Freiburg', 'Sinsheim', 'Heidelberg' and 'BadenWürttemberg', where the first three instances are all cities of Baden-Württemberg. However, the extraction of the relation type does not seem to work on German data as of yet, since the root word structure does not work for German sentences. Here a different approach is needed.

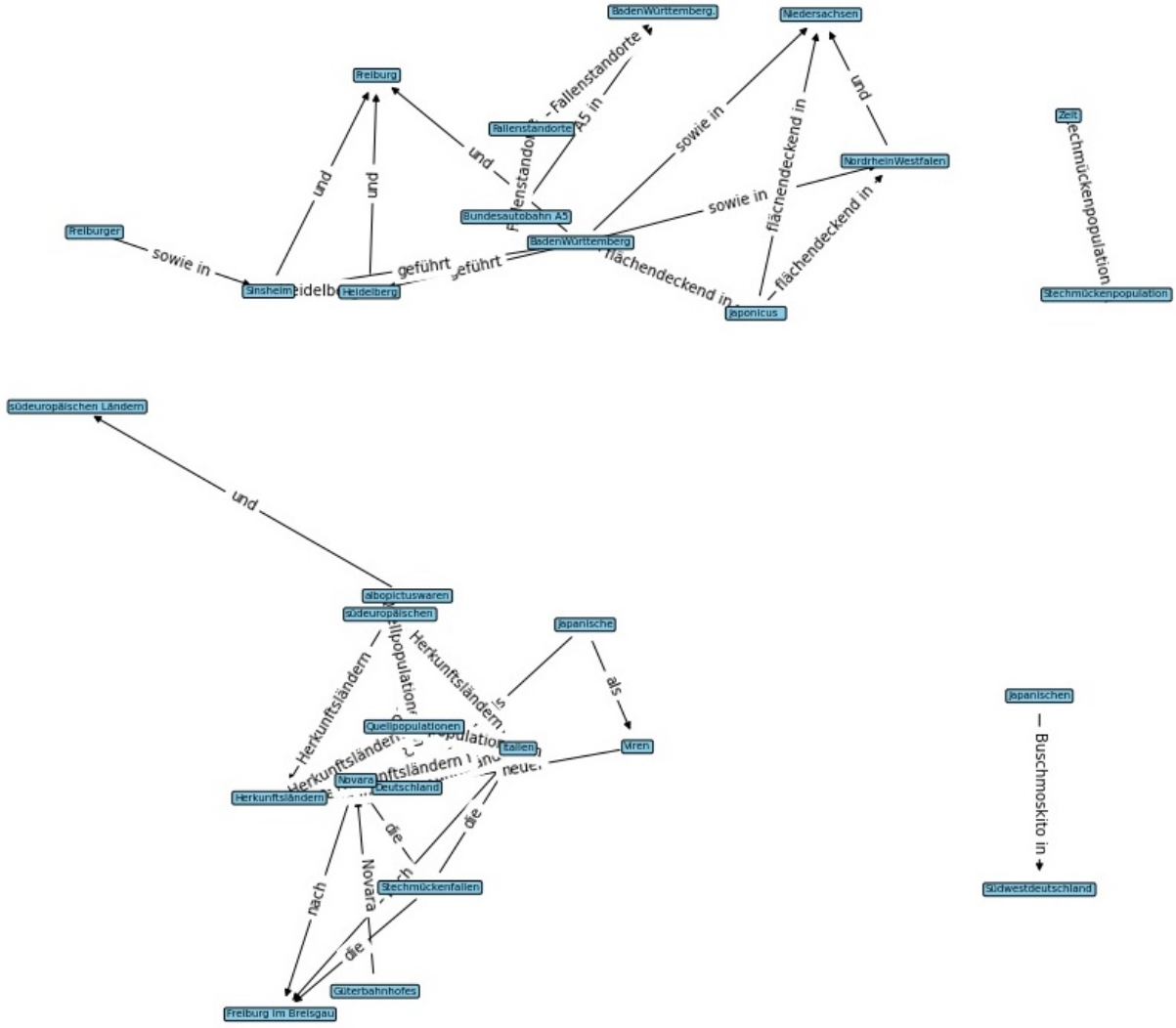


Figure 18: Root word knowledge graph for a German dataset

7 Conclusion and Discussion

Conclusion Although each part of our pipeline is researched separately for years, there is no end-to-end model available that takes PDFs or even text as input and returns the knowledge graph. In this project, we developed a full end-to-end pipeline for knowledge graph creation from a PDF file, by first defining a pipeline, with key steps needed in the process and then carefully researching each of the parts. Our final solution is able to create a concise and useful graph that can be used to identify topics and key points from a document, potentially helping to save substantial amounts of time when deployed in a business environment.
Since the main parts of the pipeline, the entity extraction model and relationship extraction model, are still an active area of research, it is especially important to use well suited models. For entity extraction, we combine two different approaches to ensure that all important entities are identified and for relationship extraction we opt for an unsupervised approach due to its flexibility and the lack of (labelled) training data. Using techniques like majority n-gram annotation, we successfully created a self-supervised pipeline that still gives useful information to the user.
In addition to identifying these models, our main contribution is to combine all parts of the pipeline. This is crucial, in order not to lose any information during the process. Any variances in the beginning of the pipeline multiply throughout every model and can lead to unusable results. Fine-tuning every part was therefore an important part of our work. It is up to future research to improve the models using more computational power and training samples. Our early experiments have shown, that the pipeline is able to extract information about the relation, but is too inconsistent to be deployed. We are optimistic that with a larger training corpus and more training time, the model will find better and finer differences between the relations.

Discussion The presented pipeline is the first step into automated document analysis, not only making masses of documents easily analyzable for humans by visualizing their contents in a graph, but also making the data readable by a computer. We will discuss further steps to improve our pipeline and also interesting research questions that can build on top of it.
Due to limitation in computing power, we were only able to train and test the models on a small subset of the available data. Our results already show the capability of the presented models, however a fully trained model will likely be more powerful and generalize better than our current best models. For example, the relationship classification model SelfORE could be trained on the full T-REx corpus [Els+18], which includes 6.2 million sentences.
The original problem formulation also included the classification of relations inside paragraphs, not only sentences. For now, we disregarded this in our pipeline, however research shows that the transformer architecture can be used for paragraph- or even document-level relationship extraction [HW20], [Zha+21]. Substituting these methods for the current relationship extraction model could expand the scope of the pipeline to this problem.
Another idea to expand the scope of the pipeline is to analyze a large number of documents at once and extract useful statistics from the identified relations. This could be done by analyzing each document with the presented pipeline and then summarizing the

results or adding them to a database, in order to calculate metrics over all results, like frequency of relations, changes in entity relations over time or semantic information between two entities.
The extracted information in the knowledge graph can also help to automatically build a knowledge database, consisting of relations and entities for future use.

Bibliography

[Age17] German Environment Agency. "Obsolescence - Political strategies for improved durability of products." In: (2017).

[Con+21] André Conrad et al. "The Zero Pollution Action Plan as a chance for a cross regulatory approach to pollution prevention and reduction." In: (2021).

[Dev+18] Jacob Devlin et al. "Bert: Pre-training of deep bidirectional transformers for language understanding." In: *arXiv preprint arXiv:1810.04805* (2018).

[Els+18] Hady Elsahar et al. "T-REx: A Large Scale Alignment of Natural Language with Knowledge Base Triples." In: *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*. Miyazaki, Japan: European Language Resources Association (ELRA), May 2018.

[Fer20] Dèlia Fernàndez-Cañellas. "TypeRE Dataset." In: (Aug. 2020). DOI: 10.6084/m9.figshare.12850154.v1.

[Gom21] Matheus Marzola Gomes. *How to Extract Named Entities from Text using Spacy Rule-Based Matching*. https://medium.com/birdie-ai/how-to-extract-named-entities-from-text-using-spacy-rule-based-matching-ef0d7e27d06c. Accessed: 2022-07-22. 2021.

[HW20] Xiaoyu Han and Lei Wang. "A Novel Document-Level Relation Extraction Method Based on BERT and Entity Information." In: *IEEE Access* 8 (2020), pp. 96912–96919. DOI: 10.1109/ACCESS.2020.2996642.

[Hu+20] Xuming Hu et al. "SelfORE: Self-supervised Relational Feature Learning for Open Relation Extraction." In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*. Online: Association for Computational Linguistics, Nov. 2020, pp. 3673–3682. DOI: 10.18653/v1/2020.emnlp-main.299.

[IBM20] Cloud Education IBM. *Natural Language Processing (NLP)*. https://www.ibm.com/cloud/learn/natural-language-processing. Accessed: 2022-07-14. 2020.

[Inc21] Alphabet Inc. *Google Knowledge Graph Search API*. 2021. URL: https://developers.google.com/knowledge-graph (visited on 07/29/2022).

[Inc04] Alphabet Inc. *Knowledge Graph search trend*. 2004. URL: https://www.google.com/trends (visited on 07/28/2022).

[Jos19] Prateek Joshi. *Knowledge Graph - A Powerful Data Science Technique to Mine Information from Text (with Python code)*. https://prateekjoshi.medium.com/knowledge-graph-a-powerful-data-science-technique-to-mine-information-from-text-with-python-f8bfd217accc. Accessed: 2022-07-22. 2019.

[Kam20] Balaji Kamakoti. *Introduction to Knowledge Graph Embeddings*. https://towardsdatascience.com/introduction-to-knowledge-graph-embedding-with-dgl-ke-77ace6fb60ef. Accessed: 2022-07-26. 2020.

[Lou20] Antoine Louis. *A Brief History of Natural Language Processing - Part 1*. https://medium.com/@antoine.louis/a-brief-history-of-natural-language-processing-part-1-ffbcb937ebce. Accessed: 2022-07-12. 2020.

[Lut21] Ben Lutkevich. *Natural Language Processing (NLP)*. https://www.techtarget.com/searchenterpriseai/definition/natural-language-processing-NLP. Accessed: 2022-07-12. 2021.

[Mat19] Ines Montani Matthew Honnibal. *spaCy meets Transformers: Fine-tune BERT, XLNet and GPT-2*. https://explosion.ai/blog/spacy-transformers#alignment. Accessed: 2022-07-29. 2019.

[Ros+22] Johanna Rose et al. "The Revision of the REACH Authorisation and Restriction System." In: (2022).

[Str+17] Emma Strubell et al. "Fast and Accurate Sequence Labeling with Iterated Dilated Convolutions." In: *CoRR* abs/1702.02098 (2017).

[TLJ20] Egbert Tannich, Renke Lühken, and Artur Jöst. "Auswirkungen des Klimawandels auf die Verbreitung Krankheitserreger übertragender Tiere (exotische Stechmücken)." In: (2020).

[Wol+19] Thomas Wolf et al. "HuggingFace's Transformers: State-of-the-art Natural Language Processing." In: *CoRR* abs/1910.03771 (2019).

[XCZ21] Wang Xu, Kehai Chen, and Tiejun Zhao. "Document-Level Relation Extraction with Reconstruction." In: *Proceedings of the AAAI Conference on Artificial Intelligence* 35.16 (May 2021), pp. 14167–14175.

[Xu+15] Yan Xu et al. "Classifying Relations via Long Short Term Memory Networks along Shortest Dependency Paths." In: *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Lisbon, Portugal: Association for Computational Linguistics, Sept. 2015, pp. 1785–1794. DOI: 10.18653/v1/D15-1206.

[Yan+21] Xi Yang et al. "Clinical Relation Extraction Using Transformer-based Models." In: *CoRR* abs/2107.08957 (2021).

[Zha+21] Ningyu Zhang et al. "Document-level Relation Extraction as Semantic Segmentation." In: *CoRR* abs/2106.03618 (2021).