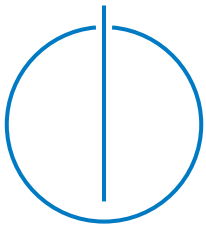
Fakultät für Informatik
Der Technischen Universität München

Master's Thesis in Data Engineering and Analytics

Automatic Norm Chain Generation for German Legal Verdicts

Jieyi Zhang



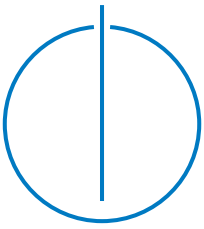
Fakultät für Informatik
Der Technischen Universität München

Master's Thesis in Data Engineering and Analytics

Automatische Generierung von Normketten für deutschsprachige Urteile

Automatic Norm Chain Generation for German Legal Verdicts

Erstbetreuer:	Prof. Dr. rer.nat. Florian Matthes
Zweitbetreuer:	Ingo Glaser, Oleksandra Klymenko
Tag der Einreichung:	15.07.2020



Declaration

I assure the single handed composition of this master's thesis only supported by declared resources.

Munich, 15. July 2020

Jieyi Zhang

Acknowledgments

I would like to thank my thesis supervisors, Ingo Glaser and Oleksandra Klymenko, from the Department of Informatics at Technical University of Munich. This work would not have been possible without their support on my research and thesis writing.

I would also like to thank Mr. Lutz Zimmer, Mr. Andre Schaper, Mr. Michael Kunze, and Ms. Anette Schimmelpfennig who participated in the validation interviews for this research project. Without their help, the survey could not have been successfully conducted.

Also, I would like to acknowledge Prof. Dr. Florian Matthes, head of Software Engineering for Business Information Systems at TU Munich. I appreciate him for this opportunity to write this thesis at Sebis chair and his time and feedback on my work.

Finally, I express my gratitude to my parents and friends for providing me with unfailing support and continuous encouragement throughout my years of master study and through the process of working on this thesis.

Abstract

In this master thesis, we developed several algorithms to automatically generate the legal norm chains for the given German verdict documents. Our dataset includes 56,606 German verdict documentations in .xml files while 32,893 files have the legal norm chains already constructed and appended by legal experts. Our goal is to use algorithms to generate the norm chains given the input text from relevant sections in the ruling documents.

Our work includes data preparation, legal norms extraction, designing and implementing the norm chain generation algorithms, and result evaluation. The algorithms here include a rule-based approach, classification models and text summarization models with neural networks. The rule-based approach selects the norms based on their frequencies and positions in the document. Since the rule-based approach only considers the current input text and always selects the existing ones from the current document, we move forward to develop machine learning and deep learning models. We tried tackling this problem as a large-scale multi-class text classification problem where each norm in the chains is a label. We then trained the models to predict if the labels need to be attached to the current case. However, there are multiple limitations while constructing the label set for the given verdict documents. Hence, we tried to apply text summarization techniques to generate the chains with more flexibility.

The results show that the rule-based approach is a strong baseline model while the other machine learning and deep learning models have their own advantages. At the end of the thesis, we discussed the potential application and future works that might be able to improve the performance of predictions.

Contents

List of Figures

List of Tables

1. Introduction

In this thesis, we first introduce the background of our project in the legal domain and the motivation behind our work. Then we set the goal of our work and list the potential challenges. In the second chapter and the third chapter, we explained the theoretical background and the related works that have been done by the other researchers. Next, we listed the research questions and the research methods for planning to solve these questions. In the next section, we presented the dataset that we used for this project and demonstrate the methodologies for legal norms extraction and norm chain generation, including machine learning and deep learning models. |n the result evaluation part, we compare the performances and analyze the benefits of different approaches. In the last chapter, we summarize this project and talked about the potential applications and future work.

1.1. Motivation

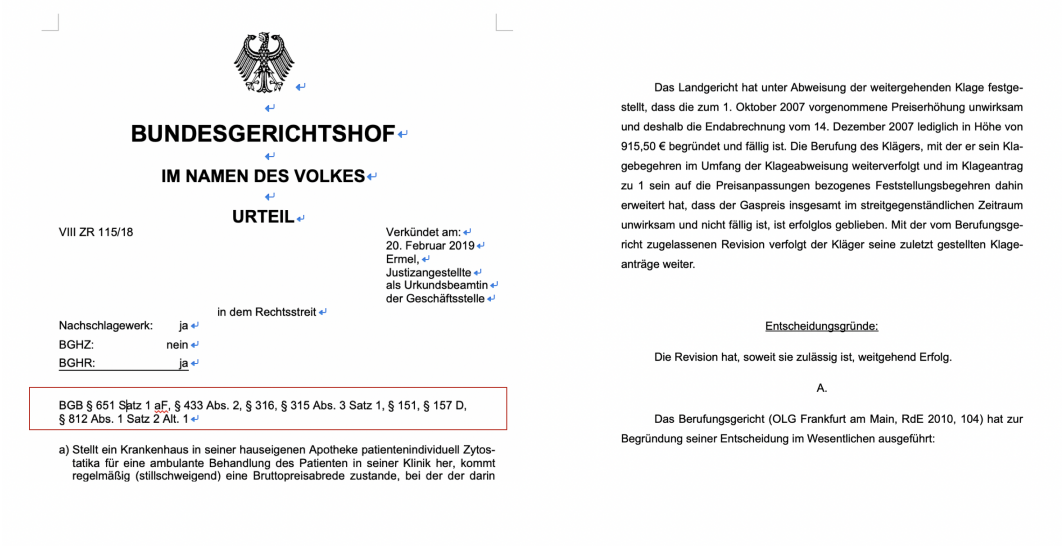
For each case in the German court, the judges and legal experts create a verdict document as a record. The format of these documents can vary from court to court. However, the major parts of the documentation are the same and are divided into different sections.

In the raw court ruling documents, the basic information, such as time and location, will be noted down at the beginning. Then the fact of the case (Tatbestand), Decision (Entscheidung), and the reasons for the judgment (Gründe) are documented in the corresponding paragraphs, and these paragraphs contain the most information of each case.

In the higher courts, such as the Federal Court of Justice (Bundesgerichtshof), these documents are normally attached to the norm chains and a guiding principle (LEITSATZ). In the cases of other courts, the norm chains and the

guiding principle will be attached usually only after the cases are published. In Fig 1.1, we can see an example of these verdict documents.

Normally, once a case is required to be published, the lawyers and legal authors will create a legal norm chain at the beginning of the document. This reference chain consists of the most important and relevant legal norms in the case and the sequence of the norms on the chain can sometimes reflect the reference relation between the norms. [OR12] This process is usually time-consuming and requires a very deep understanding of German laws. Hence, to reduce the manual work, we try to use algorithms to automate this reference chain generation.



BUNDESGERICHTSHOF

IM NAMEN DES VOLKES

URTEIL

VIII ZR 115/18

Verkündet am:
20. Februar 2019
Ermel,
Justizangestellte
als Urkundsbeamtin
der Geschäftsstelle

in dem Rechtsstreit

Nachschlagewerk: ja
BGHZ: nein
BGHR: ja

BGB § 651 Satz 1 aF, § 433 Abs. 2, § 316, § 315 Abs. 3 Satz 1, § 151, § 157 D, § 812 Abs. 1 Satz 2 Alt. 1

a) Stellt ein Krankenhaus in seiner hauseigenen Apotheke patientenindividuell Zytostatika für eine ambulante Behandlung des Patienten in seiner Klinik her, kommt regelmäßig (stillschweigend) eine Bruttopreisabrede zustande, bei der der darin

Das Landgericht hat unter Abweisung der weitergehenden Klage festgestellt, dass die zum 1. Oktober 2007 vorgenommene Preiserhöhung unwirksam und deshalb die Endabrechnung vom 14. Dezember 2007 lediglich in Höhe von 915,50 € begründet und fällig ist. Die Berufung des Klägers, mit der er sein Klagebegehren im Umfang der Klageabweisung weiterverfolgt und im Klageantrag zu 1 sein auf die Preisanpassungen bezogenes Feststellungsbegehren dahin erweitert hat, dass der Gaspreis insgesamt im streitgegenständlichen Zeitraum unwirksam und nicht fällig ist, ist erfolglos geblieben. Mit der vom Berufungsgericht zugelassenen Revision verfolgt der Kläger seine zuletzt gestellten Klageanträge weiter.

Entscheidungsgründe:

Die Revision hat, soweit sie zulässig ist, weitgehend Erfolg.

A.

Das Berufungsgericht (OLG Frankfurt am Main, RdE 2010, 104) hat zur Begründung seiner Entscheidung im Wesentlichen ausgeführt:

Figure 1.1.: An example of German verdict documents

1.2. Legislation background

Since the automatic norm chain generation is a multidisciplinary project, we did both literature research as well as interviews with the legal experts to Since the automatic norm chain generation is a multidisciplinary project, we did both literature research as well as interviews with the legal experts to understand what a norm chain is and how to create the chains for the verdict ruling document.

1.2.1. Definition of norm chain

A norm chain is a series of legal sentences that lead to the consequence of a verdict. Thus, with the help of legal norm chains, the practiced civil lawyers can save the time-consuming step of searching for the relevant and referenced legal sentences. A general definition of the legal norm chain is the following:

Definition 1.2.1. The chain of norms is a combination of explicitly or implicitly referencing legal norms that extends from a legal consequence order to the lowest level of the facts.

There is an explicit reference if the connection between two legal norms is made either by mentioning the referenced paragraph or by a term mentioned on the factual side. There is an implicit reference if the link is made only from the context and knowledge of the legal expert.

In our dataset, the candidates of legal norms in the chains are the one that appears in each document, and we assume that the importance of each norm can be reflected from the position and the frequency of each norm. However, the reference relation between each norm can be trickier to capture. Especially, the implicit reference mentioned above might be learned by the algorithms with extra data as input. For example, a referencing network between German laws might be helpful. However, in this work, we assume that the referencing can be inferred from the context and use more advanced algorithms to generate the sequence of the predicted chains.

1.3. Goal of the thesis

In our project, we are given a dataset of German verdict documentations. Among them, 32,893 files have the legal norm chains attached at the beginning of the text, while 23,713 files don't have this information. We tried three different approaches, namely a rule-based algorithm, machine learning models conducting multi-label text classification, and a sequence-to-sequence text summarization model for generating the legal norm chains.

As stated in the previous section, the correct selection of the norms is far more important for us than predicting the sequence. So one goal would be generating the norms as accurately as possible.

Besides, legal norms can have different granularities. For instance, it can be an abbreviation of law such as BGB. It can also include more details such as the paragraph number and sentence number etc. Hence, in the heuristic approach and machine learning modeling, we need to consider different granularity levels. As we will show later in the thesis, the abstractive summarization model has more flexibility in terms of predictions on different levels.

During the interview with the legal experts, they explained that the granularities of the norms in the legal norm chains depend on the complexity of the legal code. If it is a short article in the law book, then the legal code abbreviation with paragraph number might be enough. However, in the other more complicated cases, it will be much more helpful if more precise information about the clauses is given in the chains.

Hence, in this work, another goal is to construct algorithms and train models to predict the norm chains with more precise information, which means predictions with legal code abbreviations with article numbers and even some with norm numbers. The evaluation of the models will also be based on different granularity, and we try to achieve better precision with more detailed granularity.

2. Related Work

The topic of this work is novel in a way that no research has been done specifically dedicated to generating a chain of legal norms from the verdict documents. Also, not much previous work has been done in terms of generating a sequenced list of keywords based on the input text content.

However, there are still some referential methods that we can transfer to a potential solution for our scenario.

In our project, we referred to the Keyword Extraction Algorithm (KEA) developed and describe in [IWNM05] while developing the rule-based algorithm. We also used the model architectures in the work of Large-Scale Multi-Label Text Classification on EU Legislation [IC19] where they were in a much similar situation as we were. In the summarization part, we used the BERT-Transformer neural network model which is one of the most state-of-the-art algorithms in this scenario. [JD18, AV17].

2.1. Keyword Extraction Algorithm (KEA)

The process of assigning keywords to a text is called keyword indexing. To automate this process, researchers have previously developed a well-designed algorithm called Keyword Extraction Algorithm (KEA).

KEA is an algorithm for extracting keywords or key phrases from the input text. This algorithm can be used for free indexing where keywords and key phrases are selected from the original input text, or for indexing the keywords within a controlled vocabulary. [IWNM05] Normally, keywords or key phrases are multi-word units describing the content of the given text corpus. They are representative of the given text and can, therefore, provide a kind of semantic metadata that is useful for a wide variety of purposes.

In Figure 2.1, we demonstrate how this algorithm works. The pipeline takes the text from the given documents as input together with a predefined controlled vocabulary, the thesaurus. Then the algorithm processes the candidates by stemming and removing the stop words and then checks if the multi-word units from the input text match with the keyword or key phrase candidates in the thesaurus. As it is shown in the figure, the algorithm computes four features for each candidate, namely TFxIDF, first occurrence, length, and node degree. Based on these features, a model that modeling manual selection of the candidates is then used for choosing the keywords. Finally, the model computes the probabilities of the keywords and key phrases candidates and it returns the ones with top probability scores as the predictions.

In our problem setting, we analog the legal norms in the norm chain as the keywords in KEA, and used the frequency and position of each candidate norms to compute the importance of the norms in the input text and select the candidates for the norm chains.

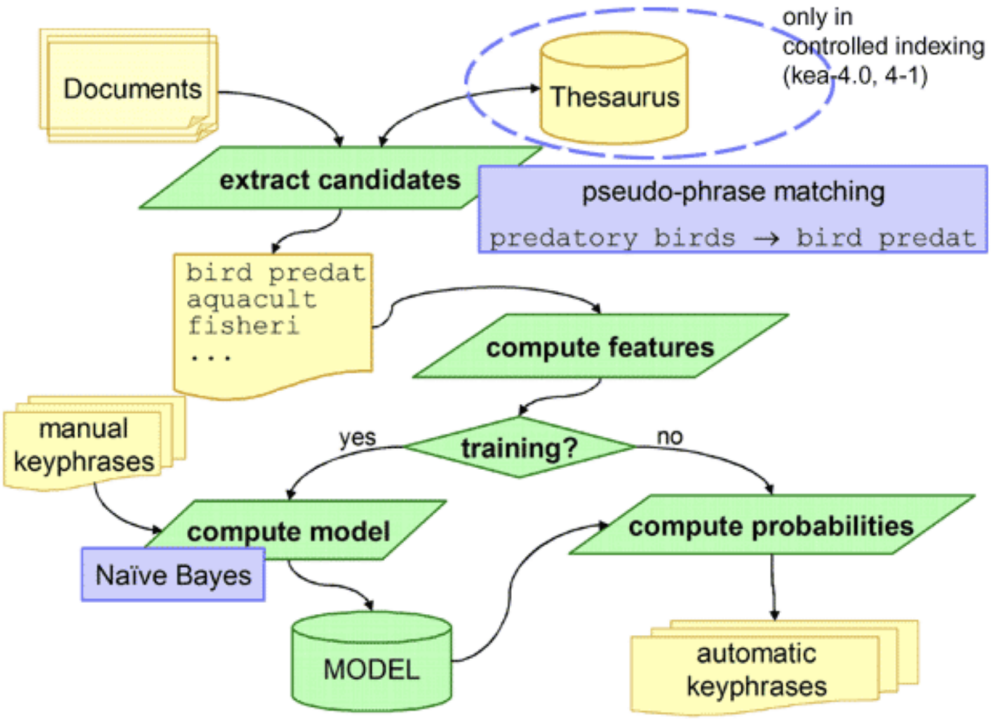


Figure 2.1.: Workflow of the KEA algorithm [IWNM05]

2.2. Large-scale multi-label text classification

In our project, we also tackled the problem by assign norms for each document from a set of legal norms of different granularity levels. This approach belongs to the multi-class text classification. However, the number of classes here varies from around 2000 to more than 10,000, depending on the granularity levels. Hence, we referred to previous work from [IC19] which is aiming for large-scale multi-class text classification in the legal-tech domain. The labels in [IC19] are the domains in the content of each document which is different from our case.

In the paper, they introduced several neural network structures that perform well with their datasets. We tried out the best two models from their suggestions with our dataset. The first one is bidirectional GRU with self-attention mechanism (BiGRU-ATT), the second one is bidirectional GRU with label-wise attention mechanism (BiGRU-LWAN).

The structure of the first model is depicted in Fig 2.2. As the implementation in paper [IC19], they used the Glove embedding model for german language to vectorize the words. The encoded part includes a layer of bi-directional GRU. A document embedding (h) is computed as the sum of the resulting context-aware embeddings ($h = \sum_i a_i h_i$), weighted by the self-attention scores a_i, and goes through a dense layer of output units with sigmoids, producing the probability for each label.

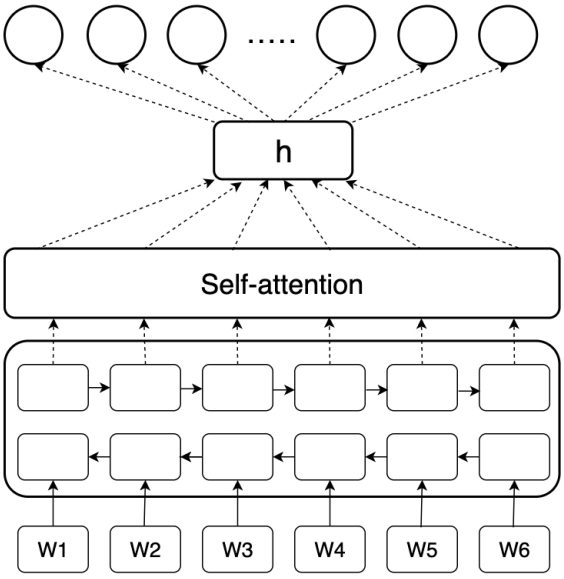


Figure 2.2.: Neural network of bi-directional GRU with self-attention mechanism [IC19]

The second model is the bi-directional GRU with label-wise attention mechanism. The structure of the second model depicted in Fig 2.3. The structure of this model is similar to the first one. The encoding part is the same as the first model and the self-attention mechanism is replaced with a label-wise attention mechanism.

Compared to BiGRU-ATT, the BiGRU-LWAN model uses L independent attention heads and each for one label. It generates L document embeddings as $\left(h^{(l)} = \sum_i a_{l,i} h_i, l = 1, \ldots, L\right)$ from the vector sequence h_i produced by the BiGRU encoder. Each document embedding $\left(h^{(l)}\right)$ predicts the corresponding label and goes through L separate dense layers with a sigmoid which returns the probability of the corresponding label.

According to the experiment results in the paper [IC19], these models perform the best in the large scale text classification problem and the performance scores are similar to each other.

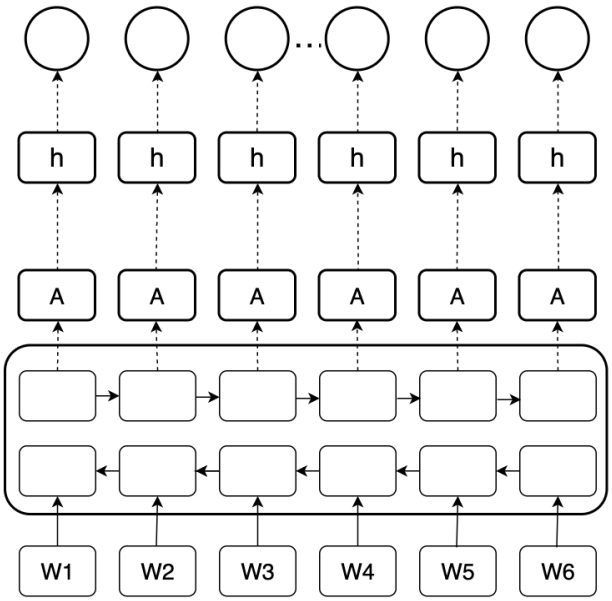


Figure 2.3.: Neural network of bi-directional GRU with label-wise attention mechanism [IC19]

2.3. BERT-Transformer abstractive text summarization

The summarization model in this thesis is the BERT-Transformer model with a pre-trained German BERT encoder. In this section, we will first introduce the Transformer network architecture developed in paper [AV17]. Then we demonstrate the BERT encoding we used in this project which is developed by [JD18].

2.3.1. Transformers

The Transformer was first proposed in the paper Attention is All You Need. Before the Transformer architecture was developed, the dominant sequence transduction models are encoder-decoder models based on complex convolutional or recurrent neural networks. Therefore, the researchers proposed the Transformer as a new simpler network architecture which solely based on attention mechanisms and entirely dispensed with convolutions and recurrence.[AV17]

The structure of the Transformer model is depicted in Fig 2.4. Basically, Transformer consists of two parts, namely encoding component, and decoding component. The encoding part is a stack of encoder layers and the decoding part is a stack of decoder layers of the same number. In the paper, the number is $N = 6$.

Each encoding layer has two sub-layers. The first is a multi-head self-attention mechanism, and the second is a position-wise fully connected feed-forward network. The encoder inputs flow through the self-attention layer which helps the encoder look at other words in the input sequence as it encodes a specific word. Then the output is fed to the second sublayer which is the feed-forward neural network.

The decoder layers include the same two sub-layers in each encoder layer as well as a third sub-layer, which performs multi-head attention over the output of the encoder stack. Also, the self-attention sub-layer in the decoder stack is modified to prevent positions from attending to subsequent positions.

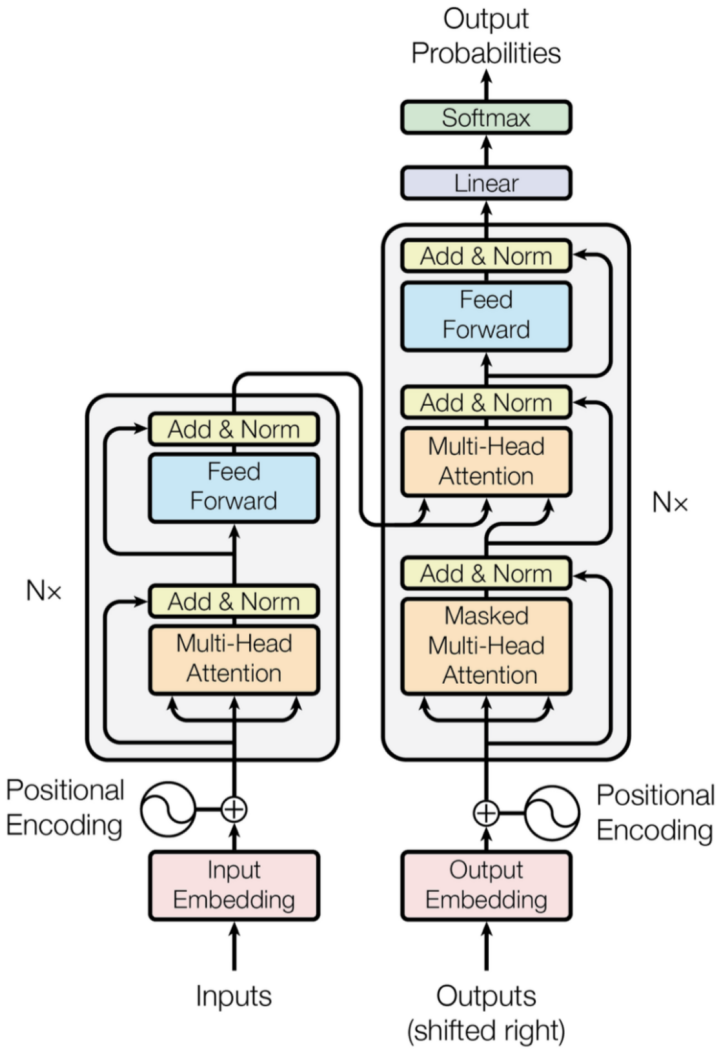


Figure 2.4.: Model architecture of the Transformer [AV17]

2.3.2. Bidirectional Encoder Representations from Transformers (BERT)

Bidirectional Encoder Representations from Transformers (BERT) is a pre-trained unsupervised language model published by Google AI Language. It is one of the state-of-the-art models and has outstanding performance in a wide variety of NLP tasks, including Question Answering, Natural Language Inference, and many others.

BERT makes use of the Transformer model we introduced in the last section. Originally, Transformer includes two separate components, namely the encoder component that reads the text input and the decoder component that produces a prediction for the task. Since the goal of BERT is to generate a language model, only the encoder mechanism is necessary. Hence, BERT model architecture is a multi-layer bidirectional Transformer encoder.

Compare to the encoding mechanism in the vanilla form of Transformer, BERT applies the bidirectional training in contrast to previous efforts that looked at a text sequence either from left to right or combined left-to-right and right-to-left training.

In the pre-training phase, two tasks are used as training strategies. First, In order to train a deep bidirectional representation, Devlin et al. mask some percentage of the input tokens at random, and then predict those masked tokens. This procedure is referred to as masked LM (MLM). The second training task is Next Sentence Prediction (NSP). In this phase, the model receives input as pairs of sentences and learns to predict if the second sentence in the pair is the following sentence in the original document. The details can be found in paper [JD18].

While the BERT language model is trained on these two specific tasks, we can still relatively easily apply it for other tasks. For each task, researchers can simply feed in the task-specific inputs and outputs into BERT and fine-tune all the parameters end-to-end.

In this work, we used the pre-training German-language BERT model trained on the latest German Wikipedia dump (6GB of raw .txt files), the OpenLegalData dump (2.4 GB), and news articles (3.6 GB). Since the training data also partly from the legal domain, we expect that this model can learn and understand the German verdict context better.

3. Theoretical Background

In this part, we explain the major theoretical background for understanding this project.

3.1. Machine learning for classification

A classification problem describes a scenario where people try to classify the data points into different groups. Statistic approaches such as machine learning algorithms are common methods for solving a classification problem. The input of the models can be continuous or discrete features, and the output is discrete representing each class in the problem setting.

The machine learning algorithms can be generally divided into three categories, namely supervised learning, unsupervised learning, and reinforcement learning.

The supervised learning models are trained with dependent outcome variables or variables which is to be predicted from a given set of predictors. Different models designed with a function that map inputs with desired outputs and the training process will optimize the prediction outcome until the model achieves a desired level of accuracy on the training data.

In contrast to supervised learning models, in unsupervised case, there is no target to estimate. For each of these models, algorithms such as the EM algorithm are designed for learning latent variables. [RS13]

Reinforcement learning models are trained to make specific decisions. During the training process, the models are exposed to an environment where it trains itself continually using trial and error.

In our project, since we have the label of the data points which are the verdict documents, we are focusing on the supervised machine learning models.

There are several most commonly used supervised machine learning algorithms for classification, including decision tree, naive Bayes, artificial neural networks(ANN), logistic regression, support vector machine and etc. In this work, we first train different types of models with their default settings, and then fine-tune the model with the best performance.

Among the algorithms from machine learning, the ANN model outperformed the other ones in our experiments and achieved the best results. The ANN is also called multi-layer perceptron which, compared to the neural networks in deep learning, has smaller and simpler structures.

3.2. Natural language processing (NLP)

Natural Language Processing (NLP) is a branch of artificial intelligence that deals with the interaction between humans using natural language and computers. There is a large variety of the objectives of NLP tasks, including translation, entity recognition, summarization and etc. The ultimate goal of the NLP models is to understand and make sense of the human languages in a manner that is meaningful and helpful. [Li01]

In this section, we are going to introduce the theoretical background for understanding NLP tasks.

3.2.1. Word embeddings

Word embedding is a method to present the words into numeric vectors and make the words with similar meaning to have a similar representation. The embedding methods can learn the vector presentation for a predefined vocabulary from a corpus of text. The training process of the embeddings can be an unsupervised process, using document statistics or joint with the neural network model on tasks like document classification.

The most commonly used embedding models include Word2Vec, GloVe, and embedding layers such as BERT.

Word2Vec, developed in 2013 at Google[TM13], is a statistical method for learning a standalone word embedding from a given text corpus. It can be trained using two methods, namely Skip Gram and Common Bag Of Words (CBOW). Both of these methods are based on neural networks. The details of the models can be found in the paper [TM13]. Basically, Word2vec takes a large corpus as its input and trained to generate a vector space with typically of several hundred dimensions. In the vector space, each unique word in the corpus will be assigned a corresponding vector. Ideally, the word vectors are located in the vector space such that words which have similar meanings or common contexts are located close to one another in the space.

Another model is GloVe. Compered to Word2vec, both models learn geometrical embeddings of words from their co-occurrence information namely how frequently they appear together in large text corpora. However, Word2vec is

a "predictive" model, whereas GloVe is a "count-based" model. The details of the model are demonstrated in the paper. [JP14]

The other type of word embedding algorithm is the embedding layer. They are a type of word embedding that is learned jointly with a neural network model on a specific natural language processing task, such as language modeling or document classification. One of them is the bidirectional Encoder Representations from Transformers.

BERT is a deeply bidirectional, unsupervised language representation, pretrained using only a plain text corpus. The advantage of BERT embedding compared to context-free models such as Word2vec or GloVe is that context-free models generate a single word embedding representation for each word in the vocabulary, where BERT take into account the context for each occurrence of a given word so that the representation can change depending on the context.

3.2.2. Text summarization

Summarization is the task of condensing a piece of text to a shorter version. Ideally, summarization reduces the size of the original text, yet it can also preserve key informational elements and the meaning of content. In general, there are two different approaches for automatic summarization: extraction and abstraction.

3.2.2.1. Extractive summarization

Extractive summarization selects sentences directly from the document based on a scoring function to form the predicted summary. The core of this method is to identify the important sections of the text and then put them together in a good format to produce a condensed version.

Hence, the extractive summarization models depend only on the extracted sentences from the original text. A typical flow of the systems consists of the following steps:

1. Generating a numeric representation of the input text with embedding methods

2. Scores the sentences based on the representation and denotes the probabilities with which they will get selected for the summary.

3. Produces a summary based on the topmost important sentences.

3.2.2.2. Abstractive summarization

The goal of abstractive summarization methods is to produce a summary by interpreting and understanding the text using advanced natural language techniques and to generate a new shorter text. Part of the generated summarization may not appear as part of the original document. Yet the condensed text should still convey the most critical information from the original text. This requires the models to be able to require and rephrasing sentences and incorporate information from full text to generate summaries. Normally, the generated text should also follow the style and logic of a human-written abstract. Thus, abstractive summarization is not restricted to simply selecting and rearranging sentences from the original text as the extractive summarization.[NRK14]

In general, building abstract summaries is a challenging task, which is relatively harder than data-driven approaches such as sentence extraction and involves complex language modeling. In this work, we used a seq2seq model BERT-Transformer neural network to summarize the input verdict documents and the summarization should ideally be the legal norm chains.

3.2.3. Sequence-to-sequence (seq-2-seq) neural network model

The seq-2-seq model was born in the field of language modeling by Google for use in machine translation. [IS14] A seq-2-seq model is a model that takes a sequence of words, letters, time series, and etc. and outputs another sequence of items.

Typically, the model is composed of an encoder and a decoder. The encoder captures the content of the input sequence and generates a hidden state vector. Then this vector is sent to the decoder, which then produces the output sequence. Since the task is sequence-based, both the encoder and decoder parts tend to use layers with LSTMs, RNNs, GRUs, and etc. The hidden state

vector is normally with a length of a larger number in the form of power of 2 so that it can effectively represent the complexity of the complete sequence as well as the domain.

The basic structure of a Seq2seq model is depicted in Fig 3.1.

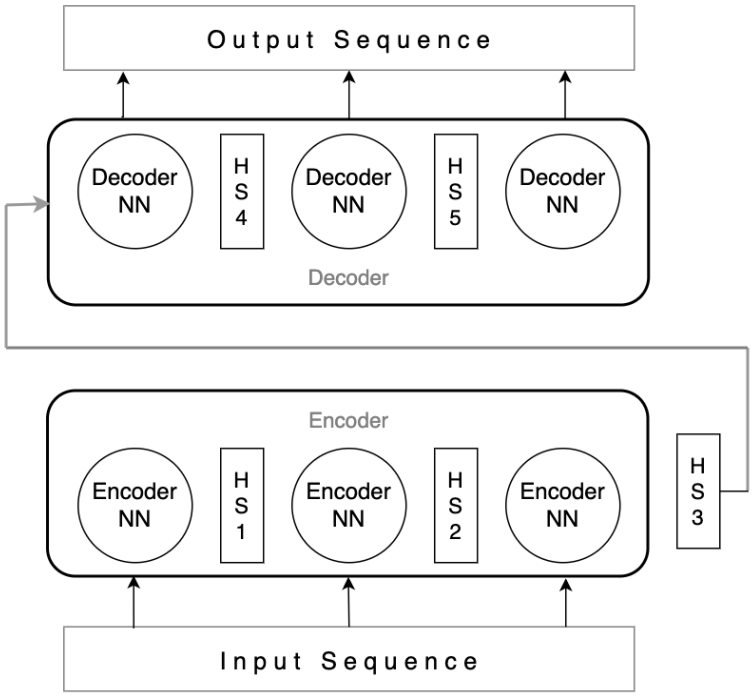


Figure 3.1.: Encoder-Decoder Model for Seq2Seq Modelling

3.2.4. Attention mechanism

The attention mechanism is added to the seq-2-seq models to improve one of its major disadvantages. The architecture of the model makes the output heavily rely on the context defined by the hidden state in the final output of the encoder. Hence, in the cases where the sentences are longer, here is a high probability that the initial context has been lost by the end of the sequence which makes it challenging for the model to deal with long input sequences.

To this end, paper [DB14] and paper [ML15] introduced and a technique called Attention which allows the model to focus on different parts of the input

sequence at every stage of the output sequence. This can preserve the context from beginning to end.

The basic idea of attention is to bring the hidden states of all the instances of the input during the decoding phase and introduce a context vector. The context vector is a weighted sum of the input hidden states as given below:

$$CV_i = \sum_k a_k \cdot HS_k$$

Where k is the number of hidden states in the encoder part.

The generated context vector is combined with the hidden state vector by concatenation and this new attention hidden vector is used for predicting the output at that time instance. As it is shown in Fig 3.2

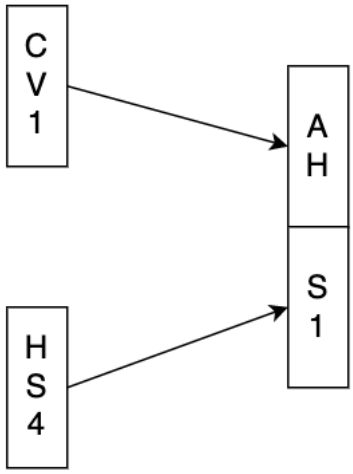


Figure 3.2.: Example of concatenation of context vector and hidden state vector of decoder layer

Finally, an alignment model that is trained jointly with the seq-2-seq model initially returns the attention scores. These scores assess how well an input represented by its hidden state matches with the previous output which is represented by attention hidden state. It conducts this matching for every input with the previous output. Afterward, these scores are fed into a softmax function and the resulting number is the attention score for each input. In

this way, the models know which part of the input is most important for the prediction.

The final model then can be simplified as the following:

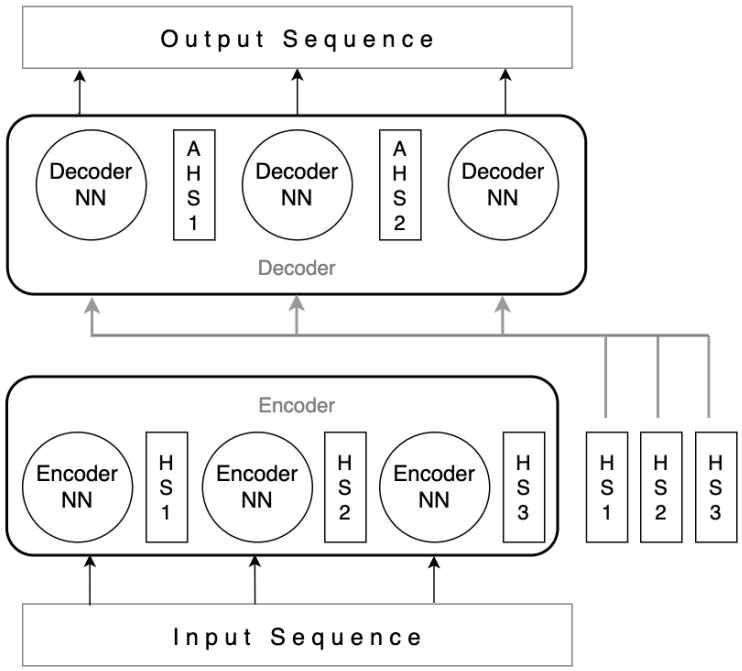


Figure 3.3.: Example of seq2seq model with attention mechanism

4. Research questions and research methods

4.1. Research questions

In this work, we are focusing on the following three major research questions:

1. How are the legal norm chains manually generated?
    a) How are the legal clauses in the norm chains selected?
    b) How to determine the sequence of the norms in the chains?
2. How to technically generate the norm chains for each verdict document?
3. Is an additional dataset required to generate the norm chains?

The first one is relevant to the legislation background, namely how are the legal norm chains manually generated. The legal norm chains include three fundamental attributes, namely the norms, the sequence, and the length of the sequence. The norms in the legal chains also have an important attribute which is its granularity level. Hence, we want to first know the manual generation process from these aspects. To investigate the first question, we checked some literature in the legal domain and talked to the legal experts, including lawyers and legal authors.

The second research question is about how to technically generate the norm chains for each verdict document. Based on the manual generation process that is described in the answer to the first research question, we first came up with the rule-based solution which is the most intuitive approach. To improved the results, we did further literature research and analogized our problem as text classification problems as well as text summarization problems. Then we tried out different algorithms to generate the legal norm chains.

The last research question that we are working on is whether norm chains can be generated just by the content in the respective verdict document. The input data of all three approaches are the verdict documents that only contain the information of each case. We assess whether an additional dataset is required to generate the norm chains based on the performances of the models as well as the information provided by the legal experts.

4.2. Research methods

The research method is a strategy used to implement a plan to answer our research questions. To investigate the questions that we have listed in the previous section, we have used the following research approaches.

1. Qualitative research methods
    a) Literature research
    b) Interviews
2. Quantitative research
    a) Statistic analysis of given dataset.
    b) Mathematical modeling and experiments to predict the norm chains

The method we used is a mix of both qualitative and quantitative methods.

Literature research is a commonly used qualitative research method. In this thesis, we did research on papers and articles of both computer science algorithm domain as well as legislation domain. Reviewing relevant previous work gave us some hint on how to solve the automatic generation problem from different approaches and also provided us with some models that are available and suitable for us to apply to our project.

The other qualitative method that we mainly used here is interview. To answer our first research question which requires background knowledge from the legal domain, we did interviews with legal experts based on our understanding from the literature research. The interviewees include lawyers, legal company partner as well as author and editor in the legal domain, and their years of

experience int he industry range from around 13 years to 25 years. The legal norm chain in the ruling files is a basic part of their daily work and they explained to us how it works in detail from the legislation perspective.

This gave us a more complete overview of the tasks that we worked on and how the legal norm chains are generated manually. Also, during the interview, we validated our results and discussed with the experienced lawyer about the performance of the models and the possible applications. This helped us to answer the third research question which is about if the current data is enough to develop an algorithm that is good enough to generate the norm chains or do we need extra input information for it.

We also applied quantitative methods including statistical analysis, modeling, and experiment. The data set we used for the quantitative research is the secondary data which are collected and annotated previously by the researchers and legal experts.

First, we conducted a statistical analysis of the original dataset. We performed exploratory data analysis mainly on the sequences and the norms in the ruling text. It helped us understand the basic features of the dataset. Then we built the models and algorithms with mathematical and computational techniques. Having the results from different approaches, we analyze the automatically generated legal norm chains from the models we developed and try to answer the second and the third research questions.

4.2.1. Interviews with legal experts

To answer the first and third research questions, we conducted three interviews with our partners in the legal domain. The exact questions and the answers are listed in the appendices. The interviews last from 40 minutes to 2 hours and were held on Zoom.

The guidelines of the semi-structured interviews are focusing on answering research questions, results validation, and application of the automation models that we have developed. We invited experienced lawyers from an international law firm and legal authors from a German legal publisher. Based on their education and experience in the industry, they answered our questions in detail

and showed us some examples from their daily work which is relevant to legal norm chains.

5. Methodology

In this chapter we will introduced three different approaches that we have used to predict the legal norm chains in our german legal verdict dataset.

5.1. German verdict document dataset

First, we present the based features of our dataset and demonstrate the results on exploratory data analysis.

5.1.1. Dataset description

The verdict documents used in our project come from more than 22 different German courts, these including:

- AG (Amtsgericht / District Court)
- AGH (Anwaltsgerichtshof / Lawyers' Court)
- ArbG (Arbeitsgericht / Labour Court)
- BAG (Bundesarbeitsgericht / Federal Labour Court)
- BFH (Bundesfinanzhof / Federal Fiscal Court)
- BGH (Bundesgerichtshof / Federal Court of Justice)
- BSG (Bundessozialgericht / Federal Social Court)
- BVerfG (Bundesverfassungsgericht / Federal Constitutional Court)
- BVerwG (Bundesverwaltungsgericht / Federal Administrative Court)
- EuG (Gericht der Europäischen Union / General Court)

- EuGH (Gerichtshof der Europäischen Union / European Court of Justice)
- EuGHMR (Europäischer Gerichtshof für Menschenrechte / European Court of Human Rights)
- FG-Pool (Finanzgericht / Finance Court)
- LAG (Landesarbeitsgericht / Regional Labour Court)
- LG (Landgericht / District Court)
- LSG (Landessozialgericht / Higher Social Court)
- OLG (Oberlandesgericht / Higher State Court)
- OVG (Oberverwaltungsgericht / Higher Administrative Court)
- SozG (Sozialgericht / Social Court)
- VerfGH (Verfassungsgerichtshof / Constitutional Court)
- VG (Verwaltungsgericht / Administrative Court)
- VGH (Volksgerichtshof / People's Court))
- Sonstige (Others)

Other basic informations about each case is also documented in the files, such as the time and the location of the case. The time span of the cases range from 2010 to 2016. Our dataset include 56,606 German verdict documentations in .xml files while 32,893 files, mading up around 58% of the total files, have the legal norm chains already generated by legal experts while being published. Within the .XML files, the format of the documents are basically the same to each other. There are nodes or elements containing the basic informations about the verdict and the ruling content.

For instance, the time of the case is documented in the node 'DATUM' and the court name is normally written in the node 'AN'. In the node 'TITEL', we can see the title of the judgement. The labels in our dataset, legal norm chains, are annotated in the nodes 'NORMENKETTE' and 'NORM'. If the case hasn't been published, normally the label will be empty or missing in the corresponding files. In some special cases, the verdict documents might have

the label norm chains only saved in the 'NORMENKETTE' node where the forms of the norms are not standardized compared to the ones in the 'NORM' nodes. Hence, in machine learning and deep learning approaches, instead of using all of the 32,893 labelled files, we only used the 32,705 files whose labels are saved properly in the 'NORM' nodes.

For the raw ruling document, namely the unpublished verdict documents, the major part of the text are the the fact of the case which is written in section 'TATBESTAND' and the reasons for the judgement in the section 'GRUENDE'. If the case is published, the legal authors and lawyers will analyze the case and append the norm chains as well as the guiding principle 'LEITSATZ' in the document.

5.1.2. Exploratory data analysis

Input sequences

First we did analysis on the length of the input sequence in the dataset. As we introduced in the previous section, there are sections where the informations are not crucial or not directly relevant to our task of chain prediction. And the highly relevant section, 'LEITSATZ', is appended together with the norm chains by the legal expert after being published. Hence, we extract the text from sections 'TATBESTAND' and 'GRUENDE' as the text input for the machine learning and deep learning models. Then we removed the special characters from the text and compute the number of words in each input sequence.

The average length of these important sections is 1,541.8 without special characters. The length distribution is plotted in Fig 5.1.

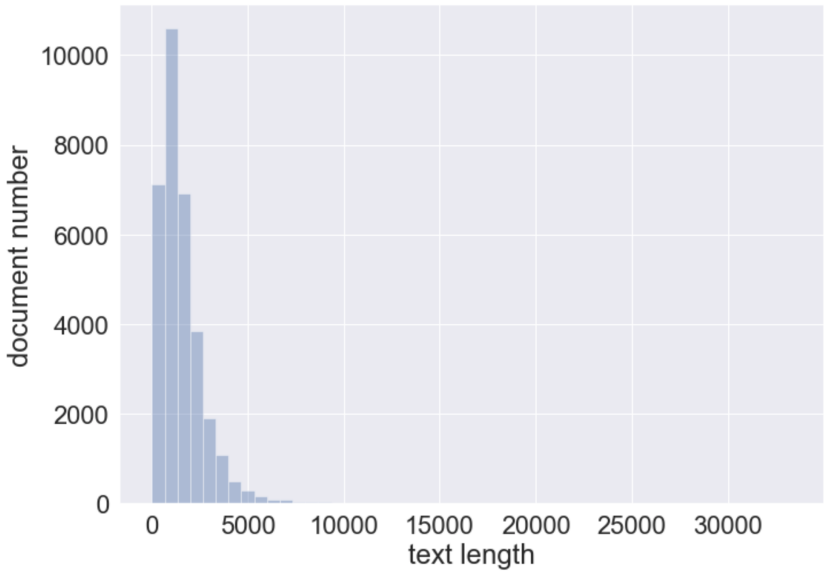


Figure 5.1.: Distribution of text length

Most documents have input sequences with less than 5,000 words. This also shows us the proportion of the text we are going to truncate if we feed these corpus into the BERT encoder which only allows sequences less than 512 tokens long.

Input legal norms

Since another type of input in our project based directly on the norms, we move forward to the analysis on the legal norms in the dataset. First we aggregate the existing norms on the abbreviation level. We observed that, there are 2,599 unique legal code abbreviations with the top frequent legal code in our dataset listed in the following table 5.1:

Norm	**Frequency**
BGB	28,766
EStG	14,852
ZPO	13,876
AO	8,337
FGO	7,338

Table 5.1.: Top frequent legal codes

Within all the abbreviations, 876 (33.71%) of them only exist once in the verdict dataset.

Then we aggregate the clauses in the dataset to the level of code abbreviation with paragraph number. This results in 16,300 combinations. With the top 5 most frequent clauses being in the following table 5.2

Norm	**Frequency**
FGO 115	1,450
BGB 611	1,450
BGB 280	1,359
BGB 823	1,276
GG 3	1,096

Table 5.2.: Top frequent legal clauses

Despite that the top norms have relevant high frequencies, there are 7,362 clauses with frequencies equal to one, making up around 45.17% of the total existing clauses. This means that when we look as the level of 'Abbr + Paragraph number', there are nearly half of the clauses only exist once in the dataset of size 32,893. In the later section where we constructed the label set of the multi-label classification task, we used this information to select the norms based on their frequencies and the total number of different norm candidates.

Norm chain labels

The average number of the norms in the labels is around 4.5 and the distribution of the label chain length is depicted in the following Fig 5.2:

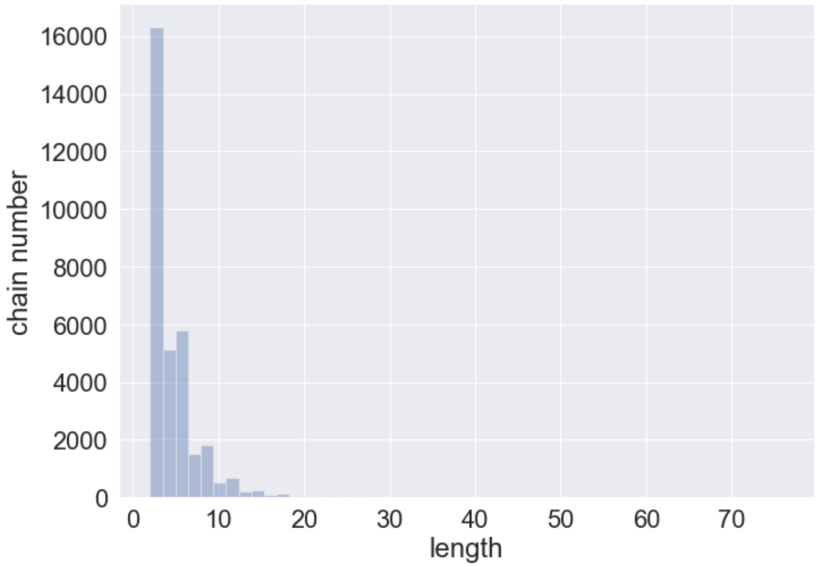


Figure 5.2.: Distribution of label length

This is the analysis results on the raw norm chain labels. In some projects under different problems settings, researchers and design algorithms to simulate the length of the chains based on the distribution and then pick the candidates. However, as the legal experts explained, the legal norms candidates are selected based on the ruling logic. Hence, we are appending norms into the labels only based on their importance in the judgment or as the algorithms predicted according to the patterns.

5.2. Rule-based algorithm

The first approach is a rule-based heuristic approach. The goal of this approach is to generate a list of norms for each input legal verdict document. In the ideal case, the element in the list shall be the ones in the label. However, since it is a process to select norm candidates from the text, the sequence of the norms cannot be predicted here.

The intuition and motivation behind this approach is that the legal norm chain of a verdict normally consists of the most important legal norms. The importance of the legal norms can be reflected by its frequency in the document and the section it occurs at. For example, the norms in the guide lines might be of more importance compared with the others. Hence, in this algorithm, we assign weights to the existing norms based on their frequency in the current document and their position. The norms which exits often and are mentioned in a more important section will be assigned with a higher weight. And algorithm selects the legal norms with higher weights into the output norm chain.

The algorithm for legal norm chain generation can be formulated in Algorithm 1.

In this algorithm, we have two types of hyper parameters, namely the weights for different sections, noted as $Weights$, and the threshold for norms selection, noted as $Threshold$. For each document, we first extract and aggregate the norms from the text in the forms of 'Law Code Abbr + Paragraph Nr' or 'Law Code Abbr + Paragraph Nr + Section Nr'. Then it counts the frequency of each norm in the current document as the weight for each norm. Norms mentioned in more crucial sections can be assigned with higher weights. Lastly, it selects the norms from the candidates set. For each group of norms that share the same 'Law Code Abbr + Paragraph Nr', it adds the norms containing section numbers with highest frequency into the norm chain. If there is no such norm candidate, it adds the 'Law Code Abbr + Paragraph Nr' part into the norm chains, if there are more than a specific number of norms in the group.

5.3. Multi-label document classification with machine learning and deep learning

The heuristic model only takes the position and frequency of the norms in each document as input, while a machine learning model normally is trained on all input training data and can learn a more comprehensive pattern.

In this part, we tackle the problem of generating a legal norm chain for each input document as a multi-label document classification problem. The labels are legal norms. Normally, the norm chains include multiple legal norms, hence it is a multi-label problem. The input of a text classification or document classification need to be in the form of numbers or vectors of numbers, therefore, a data preprocessing for the input text is required at the beginning of our machine learning pipeline.

5.3.1. Data preprocessing

While tackling the legal norm chain generation as a machine learning classification problem, we tried first feeding in the whole context into the models. To improve the results, we then extract the legal norms which appear in the context and vectorized them as another type of input for the algorithms.

5.3.1.1. Text as input

Before transferring the text into numeric vectors, we clean the input by converting the umlaut characters in German language text, removing German stop words and removing numbers with more than three digits.

Numeric presentation of text is always a challenging step in machine learning and deep learning problem. In this work, we tried multiple methods of input text vectorization, including Doc2Vec, GloVe and BERT.

First, we used the pre-trained Doc2Vec model developed by Google. [QL14] Doc2vec can create a fix-length numeric representation of a document, regardless of its text length, and the document vector intends to represent the concept of the document. A similar method to present text as vectors is GloVe. GloVe

is a word vector technique that puts words to a proper vector space, where similar words cluster together and different words repel. In the later section we will demonstrate how we applied these embedding methods to our machine learning models.

Besides, we also tried one of the state-of-art method, BERT, as the word encoder in our models. The German language model we used for our project is trained on German Wikipedia, news articles as well as German legal documents. Hence the pre-trained BERT model is proper and ready-to-use in our case. One limitation of BERT embedding is that the input text can be no longer than 512 words. Therefore, before applying BERT for word embedding, we need to truncate the input sequence to 512 words. We shortened the input by either selecting most relevant paragraphs or post truncating. Finally, we append zeros to the sequences which are shorter than 512 words.

Then the vectorized text can be feed into different models.

5.3.1.2. Extracted legal norms as input

We first experimented with Doc2Vec embedded text input with machine learning models. However, the initial results showed very poor performance of this kind of approach. The reason might be the abstractive representation of text might not be able to provide enough information to select legal norms for each case.

Another drawback of text as input of the classification models is that we need to truncate part of the input sequence. In the scenario where the input text is much longer than the threshold length, there are normally three ways of truncation, including post truncation, paragraph selection and text summarization.

First, simple truncation methods are very likely to cut off the part where there are important information for classification and result in reducing the performance.

Also, there are some models which performs well in content-based text summarizing of German language. However, one of the most important information in the verdict documents is the legal norms that exist in context. They are the

candidates of the label norm chains and should be included if summarization is required.

Therefore, in this work, we extracted the legal norms and transform them into numeric vectors by counting their frequencies and positions.

Cleaning noisy information

Although the majority of the norms follow a standard format, the code abbreviations, however, can varies a lot compared to the legal code list in official website of the German Federal Ministry of Justice. Some more complicated examples are the following:

DBA USA 1998 Art 1 Abs 1

EStG 2012 33a Abs. 1 S. 1

This means that the first part of a legal norm can not only be the code abbreviation but can also contains information such as year and location. And in some other random samples, we also saw that the abbreviation for the same code can be slightly different. Hence, we used regular expression comparing the code in the dataset to the codes in the list from Federal Ministry of Justice and filtered out the informations that doesn't match with the official list.

Norm vectorization

The flowchart of this approach is depicted in Fig 5.3. We first used regular expressions to extract all existing unique legal norms in the dataset and filter out the rare or not standard ones by comparing them with the legal norm list on the official website of Bundesamt für Justiz. Then created a vector where each legal norm is assigned with a position index in this vector and the values on the corresponding positions are the frequency of the legal norms in the current document. Also, we tried to assign different weights on the legal norms in different sections.

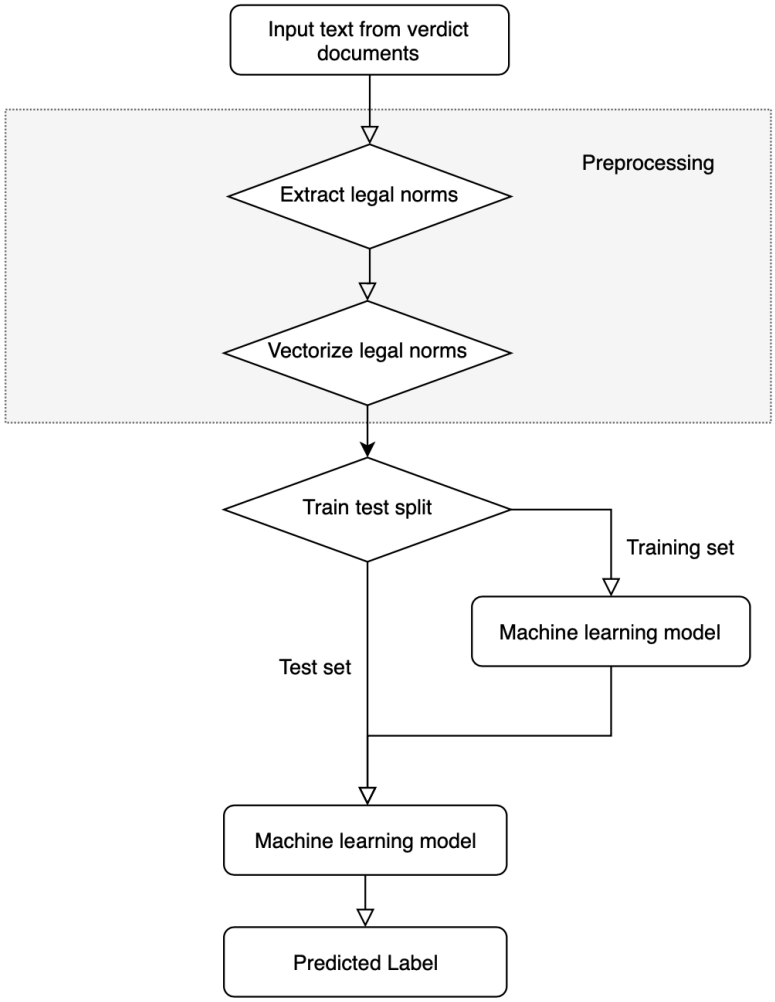


Figure 5.3.: Machine learning pipeline for verdict document classification with norm vectors as input

This will be denoted as 'NormVec' in the training and result analysis. The exact way of constructing the norm vectors is listed in the following:

1. Take the intersection set of legal code between the full list on website gesetze-im-internet.de and the ones existing in our dataset and define it as Set_{code}

2. Define the top legal code as the ones existing more than 300 time in the dataset. This threshold is tuned to a suitable size of the final output vector.

3. Scraped the sets of section numbers of the top legal code from website gesetze-im-internet.de and construct a set, $Set_{topCodeExpanded}$ of clauses in the form of 'code abbreviation + section number'.

4. The final complete set of all possible norms in the vector is $Set_{NormVec} = Set_{code} \cup Set_{topCodeExpanded}$

5. Then we construct norm vector for each input text by one-hot encoding the existing norms compared with $Sec_{NormVec}$

In the given dataset, the size of $Set_{NormVec}$, namely the length of NormVec is 22,123.

To add the position information into the numeric vectors of legal norms, extract the norms from 'TATBESTAND' and 'GRUENDE' sections and used index not only present the legal norms' frequencies but also the positions of the norms. We chose these two sections because the majority of the norms in the text are annotated in these two part. In this project, we noted this type of vectorized norm input with position information as 'NormVec w. position'.

While generating the NormVec with position, we noticed that 'code abbreviation with section number' might be a granularity level which results in a vector which is too long around with a length of 44k. Hence we skip the part of expanding the top code step compare to the generation of NormVec. The detailed steps of constructing the norm vectors with position is the following:

1. Take the intersection set of legal code between the full list on website gesetze-im-internet.de and the ones existing in our dataset and define it as Set_{code}

2. One-hot encode the norms in the 'TATBESTAND' section as vector $\mathbf{V}_{TATBESTAND}$

3. One-hot encode the norms in the 'GRUENDE' section as vector $\mathbf{V}_{GRUENDE}$

4. Then we construct norm vector for each input text by concatenating $\mathbf{V}_{TATBESTAND}$ and $\mathbf{V}_{TATBESTAND}$

In the given dataset, the length of the NormVec with postion is 1,284.

The total number of input documents is 32,705 and we split the training and testing dataset by a ratio of 7:3.

5.3.2. Model selection

In this section, we introduce the machine learning as well as deep learning models that we trained and tested and the logic behind the choices.

5.3.2.1. Machine learning models

We start with machine learning models for multi-class classification problems. We selected the models, including logistic regression, support vector machines, and multi-layer perceptron, and experimented first with the default hyper-parameters. We then compare the performances from the models and select the model with the best performances to fine tune the hyper-parameters.

The model that was selected for hyper-parameter tuning is multilayer perceptron. MLP is a class of feedforward artificial neural network. An MLP model consists of at least three layers of nodes: an input layer, a hidden layer and an output layer. The parameters including the hidden layer size of the fine-tuned models are shown in table 5.3 and 5.4.

Parameter	**Value**
activation	relu
hidden_layer_sizes	(2000,)
solver	adam
L2 penalty alpha	0.0001
learning_rate_init	0.001
beta_1	0.9
beta_2	0.999
epsilon	1e-08

Table 5.3.: Parameters of MLP classification model for norm vectors input

Parameter	Value
activation	relu
hidden_layer_sizes	(1000,)
solver	adam
L2 penalty alpha	0.0001
learning_rate_init	0.001
beta_1	0.9
beta_2	0.999
epsilon	1e-08

Table 5.4.: Parameters of MLP classification model for norm vectors with position input

5.3.2.2. Deep learning models

Since the number of different classes in this scenario is more than several thousands, we tackle this problem as an extreme multi-label text classification (XMTC). In the legal domain, there are previous work done in [IC19]. In the previous work, they developed several models for XMTC for legislation document classification. The labels in their work are the concepts in the content which are different in our case. According to the paper, the state-of-art deep learning models out-performed the baseline model which is a logistic regression machine learning approach. In our work, we selected two of the models with the best performances: bidirectional-GRU with attention mechanism and bidirectional-GRU with label-wise attention mechanism.

The architecture of these two models are shown in the figure 2.2 and 2.3 in the section Related Work. The word embedding model we used here is the German Glove model with dimension 300.

In the training data, the labels are aggregated on the 'code + paragraph number' level and only include the code which exist on the list in the website of Bundesministerium der Justiz. We filtered out the data points where there is no norms in the label and got the final dataset in the size of 31,545 files. The input sequences are truncated or post padded to the length of 1,024.

The structure and parameters of the BIGRU-LWAN model are listed in table 5.5 and the parameters of the BIGRU-ATT model are listed in table 5.6.

Layer (type)	**Output Shape**	**Param #**
InputLayer	(None, 1024)	0
Embedding	(None, 1024, 300)	300,000,600
Spatial_dropout1d	(None, 1024, 300)	0
Bidirectiona-GRU	(None, 1024, 64)	63,936
Dropout	(None, 1024, 64)	0
Labelwise_attention	(None, 10119)	1305351
Total params: 301,369,887		
Trainable params: 1,369,287		
Non-trainable params: 300,000,600		

Table 5.5.: Parameters of BIGRU-LWAN model

Layer (type)	**Output Shape**	**Param #**
InputLayer	(None, 1024)	0
Embedding	(None, 1024, 300)	300,000,600
Spatial_dropout1d	(None, 1024, 300)	0
Bidirectiona-GRU	(None, 1024, 64)	63,936
Dropout	(None, 1024, 64)	0
Seq_self_attention	(None, 1024, 64)	4,161
Flatten	(None, 65536)	0
Dense	(None, 10119)	663,168,903
Total params: 963,237,600		
Trainable params: 663,237,000		
Non-trainable params: 300,000,600		

Table 5.6.: Parameters of BIGRU-ATT model

To summarize, we use the following input-model combinations in the table 5.7

Input	**Model**
NormVec	MLP
NormVec with Position	MLP
GloVe embedded text	BiGRU-ATT
GloVe embedded text	BiGRU-LWAN

Table 5.7.: Text classification models with input

5.4. Text Summarization with BERT-Transformer model

The intuition behind this approach is that the legal norm chains are a list of most relevant and important clauses for the ruling of a case. Hence, we tried to tackle the legal norm chain generation problem as summarizing the court ruling document in the form of norm chains.

The pre-trained BERT language model is the open-source project from Deepset.ai. Deepset.ai is a startup company based in Berlin bringing their cutting-edge NLP products and services to the industry. The German BERT is one of their major products. Here are several reasons why we used the German BERT model instead of the multilingual models. The first motivation is that this model is trained on the latest German Wikipedia dump (6GB of raw txt files), the OpenLegalData dump (2.4 GB) and news articles (3.6 GB). This gave the model better ability to make sense of the individual chunks and to understand the German legal text corpus. The second reason is that this model significantly outperforms the Google multilingual model on all 5 downstream German language NLP tasks listed in the following:

1. germEval18Fine: Macro f1 score for multiclass classification - Identification of Offensive Language
2. germEval18Coarse: Macro f1 score for binary classification - Identification of Offensive Language
3. germEval14: Seq f1 score for NER
4. CONLL03: Seq f1 score for NER (deu.train - deu.testa as dev - deu.testb as test set)
5. 10kGNAD: Accuracy for document classification

Since we used the BERT encoder, the input sequence has a length limitation of 512. Hence, besides the data preprocessing in the machine learning section, we also truncated the text by taking only the 'TATBESTAND' and the 'GRUENDE' section. Also, according to the paper [JD18], we tokenized the sentence and added [CLS] and [SEP] tokens as separation of different sentences.

We kept the sequence of norms in the label and converted the list into strings. Then we compare the predicted string norm chains with the clauses in the labels. The configuration of the BERT-Transformer model is in the following table 5.8:

Model config

Parameters	**Value**
attention_probs_dropout_prob	0.1
hidden_act	gelu
hidden_dropout_prob	0.1
hidden_size	768
initializer_range	0.02
intermediate_size	3072
max_position_embeddings	512
num_attention_heads	12
num_hidden_layers	12
type_vocab_size	2
vocab_size	30000

Table 5.8.: Top frequent legal clauses

The total number of input documents is 32,705 and we split the training and testing dataset by a ratio of 7:3.

Algorithm 1: Rule-based legal norm chain generation algorithm

Result: A list of selected legal norms
D := Set of documents;
S := Set of all relevant sections in documents in D;
Weights := Dictionary with weights for different sections in S;
Threshold := Integer frequency threshold for norm candidate selection;
Chain := Output predicted list of legal norms;
for *doc in D* **do**
    FreqDict := dictionary with keys as the norms and values as their weighted frequency;
    NormDict := dictionary with keys as the legal code with paragraph number and values as a list of FreqDict whose keys share the same legal code and paragraph number;
    for *sec in S* **do**
        for *each legal norm* **do**
            Abbr := law code abbreviation;
            ArtNr := paragraph number;
            key = string(Abbr) + string(ArtNr);
            if *paragraph number exist* **then**
                ParagrNr := paragraph number;
                norm = string(Abbr) + string(ArtNr) + string(ParagrNr)
            else
                norm = string(Abbr) + string(ArtNr)
            end
            NormDict[key][FreqDict[norm]] += Weights[sec];
        end
    end
    for *(key, Dicts) in NormDict* **do**
        if *len(Dicts) == 1* **then**
            if *Dicts[norm] >= Threshold* **then**
                Chain.append(norm)
            end
        else if *len(Dicts) > 1* **then**
            norm = $\arg\max_{norm\ with\ ArtNr} Dicts[norm]$
            Chain.append(norm);
        else
            pass
        end
    end
    return Chain
end

6. Results

In this chapter, the answers to the three research questions are presented.

6.1. Answer to research question 1

The first question is about the manual generation of legal chains in German verdict documents. It includes the selection of legal norms and the sequence of them in the chains.

According to the interviewees' answers, norm chain is a small yet crucial part of the verdict ruling. To create a norm chain for a specific case, the legal experts first need to analyze in which domain the case belongs to and start with the most basic and major law code in the corresponding area. Then, the lawyers and authors go into details of the ruling and find out the crucial problems in the case. Based on the problems and questions they have listed for the case, they refer to the relevant clauses and norms with their knowledge in law. In most cases, the norm chain generation follows this logic and in the ideal cases, the ruling from the courts shall have all the relevant norms documented in the verdict documents so that the norms in the norm chains are all the existing ones from each document.

However, as the legal experts explained, there can be sloppy situations where not all the norms are mentioned in the raw ruling text or not all important and relevant norms are cited in the norm chains. Also, sometimes there can be some clauses which are too obvious, for instance, determine the amount of fine payment so that the lawyers or authors will not cite in the norm chains.

As for the sequence of the norms in the norm chains, despite that the norm chain generation follows the previously mentioned 'ruling logic' and there are reference relations between some of the norms, the sequence of the norms in the chains of the documents are not necessarily corresponding to the logic or

reference relations, according to the interviewees. Sometimes, it is even possible that the way norms are arranged to follow the alphabetic or numeric rules. Moreover, the interviewee also mentioned that compared to the sequence, the norm selection in the chains is much more important and they seldom consider the sequence in their daily work. Changing the sequence of the norms in the norm chains does not influence the information that the chains are conveying to the readers. This is one of the reasons why we focus on norm selection in this project rather than predicting the sequence of them while answering the second research question.

6.2. Research question 2

Research question 2 is about how to technically generate the norm chains for the given verdict documents dataset. We have implemented the models and algorithms mentioned in the section 'Methodology' and experimented with out dataset. The results are shown in the next several sections.

6.2.1. Result assessment metrics

6.2.1.1. Precision,recall and F1 scores

As we have discussed in the previous sections, the labels and predictions in the rule-based algorithm and text classification is a list of predicted legal norms. Hence, we assess the legal norm matching degree between the predictions and the real labels. Since the sequence of the norms cannot be predicted by these two approaches, the sequence of the norms is not taken into comparison here.

We formulated two scores to evaluate the performance of both methods. For each predicted norm chain, we need to compute how many norms in the label norm chain are captured in our predicted chain as well as how many predicted norms are actually the ones in the label norm chain. So we introduce here the two scores we have used to evaluate the results for first two methods. They are defined as the following:

$$precision\,score = \frac{\#\,of\,correctly\,predicted\,norms}{\#\,of\,norms\,in\,the\,predicted\,chain}$$

$$recall\,score = \frac{\#\,of\,correctly\,predicted\,norms}{\#\,of\,norms\,in\,the\,label\,chain}$$

This calculation can also be interpreted as :

$$\text{Precision} = \frac{true\,positive}{true\,positive + false\,positive}$$

$$\text{Recall} = \frac{true\,positive}{true\,positive + false\,negative}$$

And we compute the F_1 score from the precision and recall scores:

$$F_1 = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$$

For evaluate result on the entire dataset, we calculate the average of the scores of all precision scores and recall scores respectively.

Additionally, in the second machine learning approach, we use the standard multi-class classification evaluation metrics to assess the results.

6.2.1.2. R-Precision@K (RP@K)

The most common evaluation measures in large scale classification problems are recall (R@K), precision (P@K) at the top K predicted labels, along with micro-averaged F-1 across all labels. R@K leads to unfair penalization of methods when documents have more than K gold labels. While on the other hand, P@K leads to excessive penalization for documents with fewer than K gold labels. However, as the paper [IC19] explained, ranking measures, like R@K and P@K are sensitive to the choice of K. So they introduced R-Precision@K (RP@K) which is more suitable in these cases.

RP@K can adjust to the number of gold labels per document, without unfairly penalizing systems for documents with fewer than K or many more than K gold labels. The definition is the following:

$$RP@K = \frac{1}{N} \sum_{n=1}^{N} \sum_{k=1}^{K} \frac{\text{Rel}(n,k)}{\min(K, R_n)}$$

In effect, RP@K is a macro-averaged (over test documents) version of P@K, but K is reduced to the number of gold labels R_n of each test document, if K exceeds R_n.

Since we used the models, BIGRU-ATT and BIGRU-LWAN, from paper [IC19] and the problem settings are also similar to each other, we added this metric to assess the performance for the large scale classification approach.

6.2.2. Modelling results

In the timeline, around one month before the project deadline, we had the interviews with the legal experts. During the interviews, they pointed out one crucial point in the training dataset. According to the interviewees and our observation, section, 'Leitsatz', is also appended by the authors together with norm chains. This guiding principle section is relatively short compare to the other major sections such as the fact of cases and judgment reasoning. However, it highly concludes the case ruling and includes some of the most important legal norms.

To avoid this logical mistake, we excluded the information from this section and re-trained most of our models with the corrected dataset.

In Table 6.1, we fist listed all the results of the models trained and tested with data including information from the section 'Leitsatz'. And in the next table 6.2, we showed the corrected and re-trained results from the models.

The performance is assessed on different granularity levels, namely legal code abbreviation, abbreviation with paragraph number, and abbreviation with paragraph number as well as section (Absatz) number.

In the initial results, the performance scores of rule-based algorithm, multilayer perceptron model, bidirectional-GRU with attention neural network[1], and BERT-Transformer model are all presented. In the re-trained part, due to time limitation and the complexity of performance assessment for BIGRU-ATT and BIGRU-LWAN models, we only have the scores for rule-based algorithm, multilayer perceptron model, and BERT-Transformer model listed.

As we can see by comparing these two tables, after truncating the 'Leitsatz' section from the input dataset, the scores of machine learning and deep learning models are generally slightly lower, while the rule-based approach achieved a slightly higher performance after we fine-tuned the weight parameters in the model.

Generally, on the abbreviation and abbreviation with paragraph number level, the MLP models with norm vectors as input perform better than all the other approach. However, those models failed to predict with a reasonable accuracy

[1]The result on the legal code level was over wrtitten during re-train and missing here

on a more detailed granularity level. Compared to the MLP models, the BERT-Transformer summarization model performs better in terms of predicting on a more detailed level. The rule-based approach has the best scores on the section level and also performs well on the other more abstract granularity levels.

Prediction level		Code			Code Para			Code Para Sec		
Scores		Prec.	Recall	F1	Prec.	Recall	F1	Prec.	Recall	F1
Rule-based	Plain text	0.5489	0.6889	0.6110	0.4700	0.5813	0.5198	**0.4340**	**0.5388**	**0.4808**
MLP-2000	NormVec	0.6862	0.8017	0.7395	**0.5714**	**0.6505**	**0.6084**	NA	NA	NA
MLP-1000	NormVec w Pos.	**0.7782**	**0.8264**	**0.8016**	0.4176	0.4698	0.4421	NA	NA	NA
BIGRU-ATT	GloVe	0.7066	0.7192	0.7128	0.1655	0.2057	0.1834	NA	NA	NA
BIGRU-LWAN	GloVe	-	-	-	0.0855	0.1070	0.1128	NA	NA	NA
BERT-Transformer	Processed text	0.7486	0.6675	0.7057	0.5620	0.4474	0.4982	0.4888	0.3909	0.4344

Table 6.1.: Initial testing results on rule-based algorithm, MLP with layer size 2000 and layer size 1000, bidirectional-GRU with attention neural network, as well as BERT-Transformer model. Input data includes 'Leisatz' section.

Prediction level		Code			Code Para			Code Para Sec		
Scores		Prec.	Recall	F1	Prec.	Recall	F1	Prec.	Recall	F1
Rule-based	Plain text	0.5638	0.7440	0.6415	0.4732	0.6371	0.5431	**0.4453**	**0.5958**	**0.5097**
MLP-2000	NormVec	0.5834	0.7082	0.6397	**0.5668**	**0.6486**	**0.6049**	NA	NA	NA
MLP-1000	NormVec w Pos.	**0.6905**	**0.9427**	**0.7971**	0.4100	0.4732	0.4393	NA	NA	NA
BERT-Transformer	Processed text	0.5247	0.4781	0.5003	0.3526	0.3845	0.3679	0.3026	0.3521	0.3255

Table 6.2.: Corrected testing results on rule-based algorithm, MLP with layer size 2000 and layer size 1000, as well as BERT-Transformer model. 'Leisatz' section is excluded.

6.2.3. Rule-based algorithm

The verdict documents in our dataset are saved as .XML files. We ran the algorithm on the 32,893 verdict documents which contain the label legal norm chain at the beginning of the file. The predicted output is compared with the labels and precision scores, accuracy scores, and F1 scores are calculated. As we have explained in the methodology section, there are different granularity while comparing the norms. Hence, we compared the lists and computed the scores based on different levels, namely law code abbreviation, abbreviation with article number, and abbreviation with article number as well as norm number.

As it is shown in both initial result as well as the corrected result, this model is a relatively strong baseline on all prediction levels. Also, if we look at the 'Code Paragraph Section' level, the rule-based model always produces the best scores. However, the smaller the granularity is, namely the more information about the norms we want to predict, the lower the scores will be.

In the initial training, we got best results with the weight in 'Leitsatz' section as the highest and the variable, $threshold = 2$. After the correction in the dataset, we got best results with the weight in 'Gründe' section as the highest.

This algorithm predicts legal norm chain for each input document individually. In the next section, we will compare it with the machine learning models which learn the patterns from the information of the whole training dataset.

6.2.4. Multi-class text classification with machine learning models

6.2.4.1. MLP models with norm vectors as input

We trained models including logistic regression, naive Bayesian, and multilayer perceptron. The MLP models show better performances during the initial training experiments. Since the prediction approaches are divided into different levels and input types, we selected the best performing model, MLP, and fine-tuned it for different approaches.

Removing 'Leitsatz' in the input data results in the change of the norm vectors sizes and their values. However, the performance of the corrected model is slightly lower than that of the initial result.

The results in the tables indicate that this approach can generate so far the most accurate legal norm chains when the label set is not too large, for instance on the abbreviation and abbreviation with article number level. Yet if we take one step further and try to construct a label set with more than 10,000 labels, then the model failed to predict.

6.2.4.2. BiBRU-ATT and BiGRU-LWAN models

As we explained in the previous section, due to time limitation and complexity of retraining these two models, we will discuss the results trained on dataset including the 'Leitsatz' section.

Different from the machine learning models in the previous section, the BiBRU-ATT and BiGRU-LWAN neural network models take embedded text corpus from the documents as input. The performance on the abbreviation level is similar and slightly lower compared to the MLP models with vectorized norms as input. With the finer granularity degree of predictions and the increasing of information in the clauses, the performance of the model dropped faster than the machine learning approach and also failed to predict on the norms on the paragraph (Absatz) level.

As we introduced in the previous section, we also used RP@K to measure the performance of these two deep neural network models. We first aggregate the labels on the 'legal code' level. Here, the average norm number in the target norm chains is around 2. Hence, we compute $RP@K$ where $K = 2$ for the models. Since BiGRU-ATT model has better performance, we assess $RP@2$ on both code level and paragraph level, and the score are 0.7474 and 0.1970 respectively.

6.2.5. Text summarization with BERT-Transformer model

Similar to the MLP models, the performance of this model slightly dropped after we removed the 'Leitsatz' section from the input data.

From the result summarization table in 6.2, the summarization model doesn't achieve the highest scores in all the three granularity levels. However, compare to the heuristic approach which only select those existing clauses in the corresponding ruling text, this abstractive summarization model has the ability to generate norm chains containing clauses which do not appearing in the current input document.

More over, this approach has more flexibility in the prediction comparing to the classification models. Here is one of the prediction examples from the summarization model:

1.6 Example 6

```
[97]: sample_6 = all_df[all_df['filename']=='/Users/jieyizhang/Desktop/Master_Thesis/
      ↪labelled_files/9KSt611.xml']
      print('The label of the document '+ str(sample_6.label))
      print('The prediction of the document '+ str(pred_df.iloc[9782].pred))

      The label of the document 29615    [JVEG 5 Abs. 1, JVEG 6 Abs. 2, JVEG 19 Abs.
      2, JVEG 20, VwGO 162, VwGO 165, VwGO 151]
      Name: label, dtype: object
      The prediction of the document ['jveg 5 abs 1', 'jveg ', 'jveg 2 abs 1 s 1',
      'jveg 2 abs 1']
```

Figure 6.1.: An example comparing of the summarization model prediction v.s. the target label norm chain

Despite the fact that not all the norms in a label chain can be correctly predicted, the clauses in the predicted chain, however, have different granularity levels and are not restricted to the form of 'Abbreviation + Paragraph number + Section number' compared to the classification approach.

These are two major advantages of the summarization approach and it can be a good starting point for developing a better model with higher accuracy and more flexibility.

6.2.6. Answer to research question 2

First, if we consider the average precision, recall, and f1 scores, MLP models with norm vectors as input have the best performance on the legal code and paragraph level. The accuracy of these two prediction levels is around 0.6 to 0.7. Therefore, if our current goal to automatically generate the norm chains to the granularity of 'legal code with paragraph number', the MLP models with norm vectors as input are possible solutions.

Besides, if we try to automate the process of generating norm chains on different levels, such as sections and sentences, the rule-based algorithm and BERT-Transformer summarization model can be the options. The heuristic approach is explainable and easy to implement. The results show that this approach is relatively simpler yet also a strong baseline model in all three prediction levels. Compared to the rule-based approach, the summarization model didn't achieve high scores on the metrics. However, from the prediction examples, we can see that it can generate predictions with different granularities and even new norms that are not presenting in a specific input document. Hence, the summarization model has more potential to generate better prediction results and the models in this master thesis work can be a good starting point.

6.3. Answer to research question 3

The third research question is whether we need an additional dataset to generate the legal norm chains.

According to the answers in the interviews with legal experts, the complexity of the laws themselves is one of the factors that determine to which granularity level the norms shall be included in the chains. We compare two examples here. Fig 6.2 shows the first example of Bürgerliches Gesetzbuch (BGB) paragraph 659. Fig 6.3 shows the second example of Einkommensteuergesetz (EStG) paragraph 3.

Bürgerliches Gesetzbuch (BGB)
§ 659 Mehrfache Vornahme

(1) Ist die Handlung, für welche die Belohnung ausgesetzt ist, mehrmals vorgenommen worden, so gebührt die Belohnung demjenigen, welcher die Handlung zuerst vorgenommen hat.
(2) Ist die Handlung von mehreren gleichzeitig vorgenommen worden, so gebührt jedem ein gleicher Teil der Belohnung. Lässt sich die Belohnung wegen ihrer Beschaffenheit nicht teilen od
so entscheidet das Los.

zum Seitenanfang Datenschutz

Figure 6.2.: Content of Bürgerliches Gesetzbuch (BGB) paragraph 659

Einkommensteuergesetz (EStG)
§ 3

Steuerfrei sind

1. a) Leistungen aus einer Krankenversicherung, aus einer Pflegeversicherung und aus der gesetzlichen Unfallversicherung,
   b) Sachleistungen und Kinderzuschüsse aus den gesetzlichen Rentenversicherungen einschließlich der Sachleistungen nach dem Gesetz über die Alterssicherung der Landwirte,
   c) Übergangsgeld nach dem Sechsten Buch Sozialgesetzbuch und Geldleistungen nach den §§ 10, 36 bis 39 des Gesetzes über die Alterssicherung der Landwirte,
   d) das Mutterschaftsgeld nach dem Mutterschutzgesetz, der Reichsversicherungsordnung und dem Gesetz über die Krankenversicherung der Landwirte, die Sonderunterstützung für im Familienhaushalt beschäftigte Fraue Mutterschaftsgeld nach dem Mutterschutzgesetz sowie der Zuschuss bei Beschäftigungsverboten für die Zeit vor oder nach einer Entbindung sowie für den Entbindungstag während einer Elternzeit nach beamtenrechtlic
2. a) das Arbeitslosengeld, das Teilarbeitslosengeld, das Kurzarbeitergeld, der Zuschuss zum Arbeitsentgelt, das Übergangsgeld, der Gründungszuschuss nach dem Dritten Buch Sozialgesetzbuch sowie die übrigen Leistungen n Sozialgesetzbuch und den entsprechenden Programmen des Bundes und der Länder, soweit sie Arbeitnehmern oder Arbeitsuchenden oder zur Förderung der Aus- oder Weiterbildung oder Existenzgründung der Empfänger
   b) das Insolvenzgeld, Leistungen auf Grund der in § 169 und § 175 Absatz 2 des Dritten Buches Sozialgesetzbuch genannten Ansprüche sowie Zahlungen des Arbeitgebers an einen Sozialleistungsträger auf Grund des gesetz Forderungsübergangs nach § 115 Absatz 1 des Zehnten Buches Sozialgesetzbuch, wenn ein Insolvenzereignis nach § 165 Absatz 1 Satz 2 auch in Verbindung mit Satz 3 des Dritten Buches Sozialgesetzbuch vorliegt,
   c) die Arbeitslosenbeihilfe nach dem Soldatenversorgungsgesetz,
   d) Leistungen zur Sicherung des Lebensunterhalts und zur Eingliederung in Arbeit nach dem Zweiten Buch Sozialgesetzbuch,
   e) mit den in den Nummern 1 bis 2 Buchstabe d genannten Leistungen vergleichbare Leistungen ausländischer Rechtsträger, die ihren Sitz in einem Mitgliedstaat der Europäischen Union, in einem Staat, auf den das Abkomme Wirtschaftsraum Anwendung findet oder in der Schweiz haben;
3. a) Rentenabfindungen nach § 107 des Sechsten Buches Sozialgesetzbuch, nach § 21 des Beamtenversorgungsgesetzes, nach § 9 Absatz 1 Nummer 3 des Altersgeldgesetzes oder entsprechendem Landesrecht und nach Soldatenversorgungsgesetzes in Verbindung mit § 21 des Beamtenversorgungsgesetzes,
   b) Beitragserstattungen an den Versicherten nach den §§ 210 und 286d des Sechsten Buches Sozialgesetzbuch sowie nach den §§ 204, 205 und 207 des Sechsten Buches Sozialgesetzbuch, Beitragserstattungen nach de Gesetzes über die Alterssicherung der Landwirte und nach § 26 des Vierten Buches Sozialgesetzbuch,
   c) Leistungen aus berufsständischen Versorgungseinrichtungen, die den Leistungen nach den Buchstaben a und b entsprechen,
   d) Kapitalabfindungen und Ausgleichszahlungen nach § 48 des Beamtenversorgungsgesetzes oder entsprechendem Landesrecht und nach den §§ 28 bis 35 und 38 des Soldatenversorgungsgesetzes;
4. bei Angehörigen der Bundeswehr, der Bundespolizei, der Zollverwaltung, der Bereitschaftspolizei der Länder, der Vollzugspolizei und der Berufsfeuerwehr der Länder und Gemeinden und bei Vollzugsbeamten der Kriminalpolizei und Gemeinden
   a) der Geldwert der ihnen aus Dienstbeständen überlassenen Dienstkleidung,
   b) Einkleidungsbeihilfen und Abnutzungsentschädigungen für die Dienstkleidung der zum Tragen oder Bereithalten von Dienstkleidung Verpflichteten und für dienstlich notwendige Kleidungsstücke der Vollzugsbeamten der Angehörigen der Zollverwaltung,
   c) im Einsatz gewährte Verpflegung oder Verpflegungszuschüsse,
   d) der Geldwert der auf Grund gesetzlicher Vorschriften gewährten Heilfürsorge;
5. a) die Geld- und Sachbezüge, die Wehrpflichtige während des Wehrdienstes nach § 4 des Wehrpflichtgesetzes erhalten,
   b) die Geld- und Sachbezüge, die Zivildienstleistende nach § 35 des Zivildienstgesetzes erhalten,
   c) die Heilfürsorge, die Soldaten nach § 16 des Wehrsoldgesetzes und Zivildienstleistende nach § 35 des Zivildienstgesetzes erhalten,
   d) das an Personen, die einen in § 32 Absatz 4 Satz 1 Nummer 2 Buchstabe d genannten Freiwilligendienst leisten, gezahlte Taschengeld oder eine vergleichbare Geldleistung,
   e) Leistungen nach § 5 des Wehrsoldgesetzes;
6. Bezüge, die auf Grund gesetzlicher Vorschriften aus öffentlichen Mitteln versorgungshalber an Wehrdienstbeschädigte, im Freiwilligen Wehrdienst Beschädigte, Zivildienstbeschädigte und im Bundesfreiwilligendienst Beschädigt Hinterbliebenen, Kriegsbeschädigte, Kriegshinterbliebene und ihnen gleichgestellte Personen gezahlt werden, soweit es sich nicht um Bezüge handelt, die auf Grund der Dienstzeit gewährt werden. [2] Gleichgestellte im Sinne de Personen, die Anspruch auf Leistungen nach dem Bundesversorgungsgesetz oder auf Unfallfürsorgeleistungen nach dem Soldatenversorgungsgesetz, Beamtenversorgungsgesetz oder vergleichbarem Landesrecht haben;
7. Ausgleichsleistungen nach dem Lastenausgleichsgesetz, Leistungen nach dem Flüchtlingshilfegesetz, dem Bundesvertriebenengesetz, dem Reparationsschädengesetz, dem Vertriebenenzuwendungsgesetz, dem NS-Verfolgten

Figure 6.3.: Content of Einkommensteuergesetz (EStG) paragraph 3

We can see that BGB 659 is quite short and clear in content. So when we predicting only on level paragraph, the information it conveys is precise enough. However, if we only predict on paragraph level for EStG 3, it will be difficult for the readers to find out the exact part of text that we need to refer to. Therefore, in this case, we need to at least predict on the section level.

Hence, adding such additional information from the German laws might be helpful for improving the performance of automatic legal norm chain generation models.

Based on the scores of measuring metrics and result validation from the interviewees, we still need to achieve higher precision in order to integrate this automation tool into the legal experts' daily work. So the additional input information from an extra dataset can be a good add-on.

7. Summary and outlook

7.1. Potential applications

With the automatic norm chain generation tool that we have developed, we discussed with the legal experts about the potential applications of it.

The first straight-forward application is to generate the legal norms chains automatically. In the given dataset, there are still around half of them haven't been annotated with the norm chains. This can be a very time-consuming manual work for the lawyers. With the automated approaches, we can generate a norm chain for a specific document within seconds. It will be much faster when we use it to annotate the rest of the unpublished documents in the database.

With the fully annotated dataset, the second application could be using it for searching queries. While the lawyers and the legal authors searching for verdict cases whose ruling is based on some specific legal norms, they can retrieve more similar documents if the unpublished verdicts are also annotated. In the current website of one of the major legal domain publishers, the users can search by entering the searching bars a specific clause as a whole query or filling the legal code, article number, and other information in the corresponding sections respectively. In the case when users entering the clause as a whole query, the searching engine will parse the query and extract the code abbreviation with the article number of them. And it will search and match the ones on the same granularity level. Since our approaches generate much more accurate predictions of norm chains on the same level, this can be a good starting point for integrating the models we developed with the current searching database.

7.2. Limitations and future work

7.2.1. Limitations

A legal norm chain has three fundamental attributes, including norms, the sequence of the norms, and the length of the sequence. Within the norms, there are different granularity levels. Hence, we are analyzing the limitations of these aspects.

The first limitation is that in the rule-based and classification approaches, the models cannot capture or predict the sequence of the norms or the length of the sequence. Since the manual norm chain generation process follows a specific logic and the norm chains are not always following the ruling logic (even alphabetic sequence is possible in some cases), it was hard for us to design a heuristic approach from the ruling text. Also, the multi-label classification models are designed only for selecting the labels which are the norms in our scenario. Only the summarization model can be trained to learn the sequence pattern. Since in the beginning, we mentioned that the sequence is not of high importance in our project, we didn't focus on this part of the prediction.

The second limitation is that the majority of the models in this project doesn't have a proper mechanism to learn the granularity levels of the norms. The rule-based approach selected the norms from the input text as they are. The classification models are even stiffer due to the predefined and simplified label set of norms. In the next section, we talk about how we can improve the performance of predicting the granularity levels.

7.2.2. Future work

During the interview with some legal authors, they mentioned that the court where a specific verdict document is issued might be helpful information for modeling. Different courts have different levels and might be corresponding to different domains. In Fig 7.1, the structures of the German courts are presented.

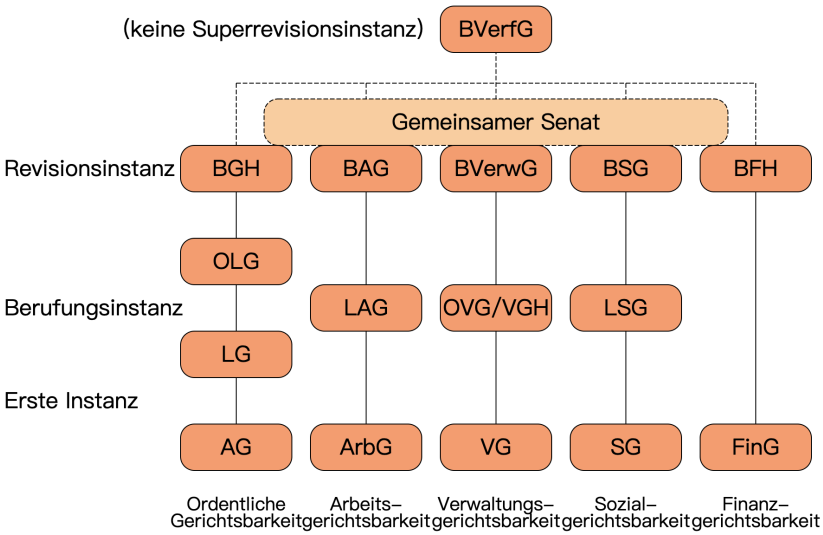


Figure 7.1.: Courts in Germany position in different levels and domains. Resource: C.Löser

For example, if in some court their specialization includes tax law, then the granularities of the norms in the norm chains of its documents are normally on a more detailed level. The reason behind this is that tax law content is usually longer and more complicated. Each chapter might include a larger number of sections subsections and etc. Hence, the lawyers need to cite the clauses with more precise information.

Besides adding more features or information into the models, we can also try more advanced models in the future. For instance, there are text summarization models that consist of several sub-models. One of them can learn the structure of the summarization. This might be helpful to improve the accuracy of the predictions from the summarization neural network models.

7.3. Summary

To conclude, in this master thesis, we developed several algorithms to automate the generation of the legal norm chains in the German verdict documents. The goal is to generate the chain with the correct legal norms for each given input document. The algorithms include both heuristic approaches as well as multi-label classification models and text summarization neural networks.

In the heuristic approach, we used the frequencies and positions of the legal norms in the text to represent their importance in the ruling document. The algorithm selects the most important ones from the candidates and forms the norm chains.

In the classification method, we tried to solve the problem with machine learning models and deep learning neural network. To provide more precise and complete input information for the machine learning models, we generate the numeric vector presentation for the norms in the input text. The labels are the norms in the target chain and we train the models for the multi-label classification task. Also, we applied the neural networks developed in [IC19] for classifying documents with a large label set. The input is text embedded by the GloVe language model.

In the summarization approach, we used the pre-trained BERT German language model with a Transformer. We set the target legal norm chains as strings of the summarization for each input text and train the model to predict the chains by summarizing the text.

The assessment of the models is conducted on different granularity levels, namely predicting the clauses with code abbreviation level, abbreviation and article number level, or even with paragraph number. As it is shown in the results, the rule-based approach performs generally well on different levels while the machine learning classification approach achieved a higher accuracy on more abstractive levels of predicted chains. However, the drawback of the classification approach is that it failed to predict with more detailed information. The summarization model, on the other hand, has more flexibility in terms of predicting legal clauses on different levels and can predict more information in the norm chains.

The performance of the predicting models can be further improved by changing model structures or providing more input information for the training procedure. With an automatic norm chain generation tool, the legal experts can easily annotate the rest unpublished verdict documents. These information can be very helpful for the users to search for relevant documents in the database.

Appendices

A. Glossary

Abs. Absatz (section)
BIGRU-ATT Bidirectional GRU with Self-attention Neural Network
BIGRU-LWAN Bidirectional GRU with Label-wise-attention Neural Network
BERT Bidirectional Encoder Representations from Transformers
BGB German Civil Code
GloVe Global Vectors for Word Representation
KEA Keyword Extracting Algorithm
MLP Multi-Layer Perceptron
NLP Natural Language Processing
Nr. Nummer
TP True Positive
FP False Positive
TN True Negative
FN False Negative
S. Satz (sentence)

B. Interview guide

Interviewee background

1. What is your field of working?
2. What is your current position or job title?
3. How many years of experience do you have?
4. How would you assess your knowledge on the legal norm chains?

How are the legal norms in the norm chain selected manually?

1. Are they always selected from the ones in the text of the verdict documents? If not, which kind of extra documents/references the legal experts normally need?
2. How are the norms selected manually? Currently, we use their position and frequency. Does it make sense?
3. How to determine the sequence of the norms?
4. How to determine the granularity of each norm? Is a more detailed or more abstract legal norm preferred?
5. Regarding to the uniqueness of a chain, are there many possible solutions?
6. When and how often the chains are not added by the court (i.e. when are they missing?)

In our work, we generate the predicted norm chains for each verdict document on the following granularity: 'Legal code', 'Legal code + Paragraph number' and 'Legal code + Paragraph number + Section number'

1. The results so far suggests that if we predict on a 'Legal code' or 'Legal code + Paragraph number' level, where the algorithms achieve better accuracy/precision scores, if these predictions will be helpful from a legislation perspective?

2. Is it possible to have any inclusive relation between different norms?

More on the results: questions and suggestions:

1. How can we integrate the automatic generation of legal norm chains into work?

2. Which other potential applications do you see?

C. Interview transcript

C.1. Interview 1

Interviewee background

- Field of working: Legislation
- Current position or job title: Of counsel at Baker McKenzie
- Years of experience in this industry: 22+ years
- Knowledge on the legal norm chains: Expertised

How are the legal norms in the norm chain selected manually?

- Are they always selected from the ones in the text of the verdict documents? If not, which kind of extra documents/references the legal experts normally need?
  - Answer: Ideally, the raw ruling documents should contain all the relevant norms for each case, so principally we only need to select those ones existing in the current text. However, sometimes there can be sloppy cases where the norms are not included in the text then we need to add the extra norms to the chains. The norms are all from the law codes.
- How are the norms selected manually? Currently, we use their position and frequency. Does it make sense?
  - Answer: We first analyze the case and define the critical problems and questions for each case based on the legal knowledge and training for lawyers. For each problem, we then list the legal norms that are relevant to it and finally add them together into the norm chain.

Position can be useful in identifying the norms' importance. For instance, the norms in reasoning section are normally more crucial for a case. The frequency can also be helpful, but it is not always the case that the important ones will be repeated several times.

- How to determine the sequence of the norms?
  - Answer: The sequence is not a very important feature in the norm chains. One way of putting norms into a sequence is following a logic sequence. But we seldom consider the sequence of them. In some cases, however, I have also seen alphabetic sequences.
- How to determine the granularity of each norm? Is a more detailed or more abstract legal norm preferred?
  - Answer: It depends on the norms and the problems. But generally, we prefer a more detailed reference, because we can provide more information to the readers.
- Regarding to the uniqueness of a chain, are there many possible solutions?
  - Answer: In the ideal case, the set of norms in the chain of a verdict case should be unique. However, as we talked about before, there are some sloppy cases and human beings cannot always be 100% correct. So in the practice, there are different solutions.
- When and how often the chains are not added by the court (i.e. when are they missing?)
  - Answer: If the case is handled in the BGH which is the highest court, then normally they append the norm chains in the raw ruling documents.For the other courts, normally the norm chains are appended by lawyers after a case is processed for publishment.

Prediction granularity

- The results so far suggests that if we predict on a 'Legal code' or 'Legal code + Paragraph number' level, where the algorithms achieve better accuracy/precision scores, if these predictions will be helpful from a legislation perspective?

  - Answer: If only predicting the legal code, it cannot give us enough information about the references being used. For example, if we actually check some of the law code books, they contains different chapters and paragraphs. Hence, if we only have the code, we cannot find the exact information we are looking for.

- Is it possible to have any inclusive relation between different norms?

  - Answer: In the norm chains, there should be no inclusive relation between different norms. And as we said before, we prefer more detailed predicted norms.

More on the results: questions and suggestions:

- How can we integrate the automatic generation of legal norm chains into work?

  - Answer: For example, if we want to retrieve documents which contains specific norms, then we can search in the database with the automatic annotated norm chains.

- Which other potential applications do you see?

  - Answer: Currently, we have only consider adding these norm chain feature into the legal verdict database. Probably there will be some other applications in the future.

C.2. Interview 2

Interviewee background

- Field of working: Legislation
- Current position or job title: Leader of electronic media at Otto-Schmidt
- Years of experience in this industry: 25+ years
- Knowledge on the legal norm chains: Expertised

How are the legal norms in the norm chain selected manually?

- Are they always selected from the ones in the text of the verdict documents? If not, which kind of extra documents/references the legal experts normally need?
  - Answer: Normally this should be the case, since the ruling documents should containing all the legal norms that are relevant to each case.
- How are the norms selected manually? Currently, we use their position and frequency. Does it make sense?
  - Answer: The norms are selected based on the analysis from the lawyer. Based on the analysis results and their knowledge in this domain, they will select the most clauses that need to be included in the chains.
- How to determine the sequence of the norms?
  - Answer: The sequence of the norms within the same code can follow the increasing paragraph numbers. In the other cases it might follow the ruling logic. As you have seen in some other cases, there are some 'combinations' of legal norms that usually appear together in the documents and their co-occurance represent some specific information. However, there is not a very clear rule for the norm sequence in the norm chain.
- How to determine the granularity of each norm? Is a more detailed or more abstract legal norm preferred?

  - Answer: The granularity depends on the complexity of the code that we are referring. If the code has very long and detailed content, then we need to add more details about which part of the code is relevant. Similarly, on the code with paragraph level, if there are many sections in this paragraph, then we need to write down the exact number of the sections. A more detailed norm chain is preferred, since we can always extract the abstractive information from them.

- Regarding to the uniqueness of a chain, are there many possible solutions?
  - Answer: Since the selection of the norms is based on the analysis from the lawyers processing the documents, the result might be different from lawyer to lawyer.
- When and how often the chains are not added by the court (i.e. when are they missing?)
  - Answer: The norm chains are usually missing for the unpublished cases. Then the authors will append the chains and the 'Leitsatz' section together to the raw documents.

Prediction granularity

- The results so far suggests that if we predict on a 'Legal code' or 'Legal code + Paragraph number' level, where the algorithms achieve better accuracy/precision scores, if these predictions will be helpful from a legislation perspective?
  - Answer: As we talked about before, we prefer more precise and detailed predictions. However, if we use our current search engine to look up the documents based on the norms in norm chains in our database, the search queries will always be aggregated on the 'code with paragraph number' level. So, for now, 'code with paragraph number' level is also fine for us.
- Is it possible to have any inclusive relation between different norms?

  - Answer: There shouldn't be such cases, if in the prediction there is a potential inclusive relation between different norms, we would suggest to keep the more detailed ones.

More on the results: questions and suggestions:

- How can we integrate the automatic generation of legal norm chains into work?
  - Answer: For now, we would like to apply it for searching in our database. If we use the automatic tool annotating all the ruling documents, we can search for the cases with the clauses in the norm chains. (More demos on how the search works in their website)
- Which other potential applications do you see?
  - Answer: Using it for the searching the documents in our database is the major application for us now.

C.3. Interview 3

Interviewee background

- Field of working: Legislation
- Current position or job title: Editor in tax editorial at Otto-Schmidt
- Years of experience in this industry: 13+ years
- Knowledge on the legal norm chains: Expertised

How are the legal norms in the norm chain selected manually?

- Are they always selected from the ones in the text of the verdict documents? If not, which kind of extra documents/references the legal experts normally need?
  - Answer: We either select it from the norms in the verdict documents or based on our analysis we add the norms that are highly relevant.
- How are the norms selected manually? Currently, we use their position and frequency. Does it make sense?
  - Answer: The norms are selected based on the content of the case and if we think they are crucial for the judgement. Theoretically, we need to put all the norms which are relevant based on our analysis, however, the chains should not be too long. In such cases, we need to truncate the chains to a proper length.
- How to determine the sequence of the norms?
  - Answer: The norms can be sorted based on the alphabetic sequence or numeric sequence. However, if some clauses are significantly more important than the others, we will put those in the beginning of the chains.
- How to determine the granularity of each norm? Is a more detailed or more abstract legal norm preferred?

  - Answer: It depends on the ruling text and which part of the code we are referring to. If the ruling text is very abstract, then we only give the corresponding clauses on an abstract level. Basically, we need to tell the readers which part of the law we refer to and we generally prefer to give the norms as precise as possible.

- Regarding to the uniqueness of a chain, are there many possible solutions?

  - Answer: Switching the norms in the norm chains normally doesn't effect the meaning it conveys. Hence, if we consider the sequence, for instance, there are multiple solutions.

- When and how often the chains are not added by the court (i.e. when are they missing?)

  - Answer: In recent years, most of the verdict documents have already had the norm chains appended from the courts. Previously, for the cases which were not published, the norm chains might be missing.

Prediction granularity

- The results so far suggests that if we predict on a 'Legal code' or 'Legal code + Paragraph number' level, where the algorithms achieve better accuracy/precision scores, if these predictions will be helpful from a legislation perspective?

  - Answer: As a legal publisher, we really rely on very precise norms. If there are only informations about the legal code, for instance, the norms will be too blurry for us. But so far our search engine will also aggregate the queries on the 'Legal code with Paragraph number' level, so this granularity should be fine for us now.

- Is it possible to have any inclusive relation between different norms?

  - Answer: No, as we said before, we tend to give readers very precise norms, so in such case we will use the norms will more details.

More on the results: questions and suggestions:

- How can we integrate the automatic generation of legal norm chains into work?
  - Answer: If the information that is predicted is very blurry, then currently we cannot directly apply such automation into our work of generating norm chains.
- Which other potential applications do you see?
  - Answer: This can be a good starting point for the automation of norm chain generation. Later when we have more precise prediction, we can use it to automate the process and add the generated information into our dataset.

Bibliography

[AV17] A. Vaswani, N. Shazeer, N. P. J. U. L. J. A. N. G. L. K.: *Attention is all you need.* 2017. (document), 2, 2.3, 2.3.1, 2.4

[DB14] D. Bahdanau, K. Cho, Y. B.: *Neural Machine Translation by Jointly Learning to Align and Translate.* 2014. 3.2.4

[IC19] I. Chalkidis, M. Fergadiotis, P. M. N. A. I. A.: *Extreme Multi-Label Legal Text Classification: A case study in EU Legislation.* 2019. (document), 2, 2.2, 2.2, 2.3, 5.3.2.2, 6.2.1.2, 7.3

[IS14] I. Sutskever, O. Vinyals, Q. V. L.: *Sequence to sequence learning with neural networks.* 2014. 3.2.3

[IWNM05] I. Witten, G. Paynter, E. F. C. G.; Nevill-Manning, C.: *KEA: Practical Automated Keyphrase Extraction.* 2005. (document), 2, 2.1, 2.1

[JD18] J. Devlin, M. W. Chang, K. L. K. T.: *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding.* 2018. 2, 2.3, 2.3.2, 5.4

[JP14] J. Pennington, R. Socher, C. D. M.: *GloVe: Global Vectors for Word Representation.* 2014. 3.2.1

[Li01] Liddy, E. D.: *Natural Language Processing.* 2001. 3.2

[ML15] M. Luong, H. Pham, C. D. M.: *Effective Approaches to Attention-based Neural Machine Translation.* 2015. 3.2.4

[NRK14] N. R. Kasture, N. Yargal, N. N. S.: *A Survey on Methods of Abstractive Text Summarization.* 2014. 3.2.2.2

[OR12] Oliver Raabe, Richard Wacker, D. O. C. B.: *Recht ex machina: Formalisierung des Rechts im Internet der Dienste.* 2012. 1.1

[QL14] Quoc Le, T. M.: *Distributed Representations of Sentences and Documents*. 2014. 5.3.1.1

[RS13] R. Sathya, A. A.: *Comparison of Supervised and Unsupervised Learning Algorithms for Pattern Classification*. 2013. 3.1

[TM13] T. Mikolov, I. Sutskever, K. C. G. C. J. D.: *Distributed Representations of Words and Phrases and their Compositionality*. 2013. 3.2.1