Summarization of German Court Rulings

Ingo Glaser[*] and **Sebastian Moser**[*] and **Florian Matthes**
Technical University of Munich
ingo.glaser@tum.de
sebastian.moser@in.tum.de
matthes@in.tum.de

Abstract

Historically speaking, the German legal language is widely neglected in NLP research, especially in summarization systems, as most of them are based on English newspaper articles. In this paper, we propose the task of automatic summarization of German court rulings. Due to their complexity and length, it is of critical importance that legal practitioners can quickly identify the content of a verdict and thus be able to decide on the relevance for a given legal case. To tackle this problem, we introduce a new dataset consisting of 100k German judgments with short summaries. Our dataset has the highest compression ratio among the most common summarization datasets. German court rulings contain much structural information, so we create a pre-processing pipeline tailored explicitly to the German legal domain. Additionally, we implement multiple extractive as well as abstractive summarization systems and build a wide variety of baseline models. Our best model achieves a ROUGE-1 score of 30.50. Therefore with this work, we are laying the crucial groundwork for further research on German summarization systems.

1 Introduction

Most modern summarization systems are built around a handful of datasets predominantly collected from newspapers. In recent years many researchers saw the need to introduce different datasets to the summarization domain such as ArXiv/PubMed (Cohan et al., 2018), XSum (Narayan et al., 2018), or datasets from the legal domain like BigPatent (Sharma et al., 2019) and BillSum (Kornilova and Eidelman, 2019). As the performance of data-driven systems is highly impacted by the quality and size of the dataset used, we introduce a new dataset for the German legal domain, more specifically a collection of court rulings.

Court rulings are long documents clarifying legal interpretations in a matter of dispute. Due to their complexity and length, it is of critical importance that legal practitioners can quickly identify the content of the verdict and thus decide on the relevance for their legal case.

German court rulings have three basic parts. First, the tenor denotes the explicit decision a judge has made. Second, the main body of a judgment is divided into the facts and the reasoning, explaining why the judge came to their conclusion given the specific preconditions. Third, the rubrum contains meta-information, i.e. previous instances, the most important legal texts referenced, and the guiding principle. The guiding principle shortly summarizes why the specific decision was made and therefore is extremely useful for legal practitioners. This high-level summarization allows them to quickly assess whether the court ruling is important for them or not.

Not all of those pieces of information are found in all verdicts. Especially the guiding principle is only added by the judge or a legal publisher if the judgment contains an important legal decision. A lot of expert knowledge and time is necessary to create this summarization, which is why we want to automate this task. Therefore, with this paper our contributions are the following:

1. A large scale collection of German court rulings for summarization with 100k verdicts
2. A pre-processing pipeline specifically tailored to German court rulings
3. Multiple extractive as well as abstractive summarization models tailored to our dataset

All models described here are implemented in PyTorch. All code for this paper is openly available on GitHub[1].

[*] Equal contribution

[1] at github.com/sebimo/LegalSum

2 Related Work

The literature differentiates two base approaches to text summarization: (1) extractive and (2) abstractive summarization. In the extractive case, sentences or tokens are copied and extracted from the given text and form a summarization. Contrary, the abstractive summarization system is creating a summary word by word, thus it is not dependent on the words and information used in the verdict. This is why abstractive summarization is often referred to as natural language generation (NLG).

Most commonly, extractive summarization systems use a hierarchical document representation, i.e. words are combined into sentences, sentences are aggregated into documents. Various different architectures with convolutional neural networks (CNN), recurrent neural networks (RNN), and attention mechanisms (Nallapati et al., 2017; Cheng and Lapata, 2016; Jadhav and Rajan, 2018) are used to extract the most important sentences. More recent approaches focus on transformer architectures (Liu and Lapata, 2019; Zhang et al., 2019; Xu et al., 2020; Zhou et al., 2020) and investigate the level of extraction (Xu et al., 2020; Zhou et al., 2020; Jadhav and Rajan, 2018) by selecting words or n-grams instead of whole sentences.

The most common way to tackle abstractive summarization is to use an encoder-decoder structure. Hereby the encoder is aggregating the information from the text and the decoder is a conditional language model producing the words in the summary. Hierarchical document representations are again important and similar modeling approaches are used (Nallapati et al., 2016; Gehrmann et al., 2018; Li et al., 2018; Chopra et al., 2016). In abstractive summarization, there is also a shift towards transformers in recent years (Liu and Lapata, 2019; Dong et al., 2019; Qi et al., 2020; Lewis et al., 2020; Raffel et al., 2020; Zhang et al., 2020).

Many different techniques were found to be useful for summarization. Pointer Networks (Vinyals et al., 2015) allow the selection of words from the input text and are used in extractive (Jadhav and Rajan, 2018) as well as abstractive summarization (See et al., 2017). Some papers use templates or salient sentences in the word generation process (Cao et al., 2018; Bae et al., 2019; Chen and Bansal, 2018). Others try to discourage the system from repeating certain phrases by introducing special loss function such as coverage (See et al., 2017) or novelty (Kryściński et al., 2018).

Especially long documents are challenging and thus multiple approaches try to tackle this problem e.g. by introducing multiple encoders to find a hierarchical representation of the text (Celikyilmaz et al., 2018) or introducing the discourse structure into a hierarchical attention model (Cohan et al., 2018).

3 Dataset

3.1 Dataset acquisition

The dataset consists of verdicts from the Dr. Otto Schmidt publisher and judgments which were scraped from gesetze-bayern.de and justiz.nrw.de. All those verdicts were then automatically translated to a unified JSON format. It contains the meta-information about the verdict: id (unique identifier), date, court, normchain (the most important legal texts referenced), all the referenced norms, the previous instances, mentioned keywords, and verdict title. All the referenced norms are stored with a unique placeholder (__normxyz__), which is used in the text body of the verdict to quickly identify a norm during training. Norms and their placeholder are stored with the verdict and the norm can be reintroduced later on. Additionally, the text segments of the verdict (tenor, facts, reasoning, guiding principle) are stored sentence by sentence, as the sentence segmentation is non-trivial for German legal documents as identified by Glaser et al. (2021). We used their model to automatically segment the verdicts into sentences.

Not every piece of information was directly accessible in the original data formats. For example, the HMTL format of justiz.nrw.de does not contain any annotations regarding referenced norms in the texts, thus it was necessary to extract them. In other cases, the main body of the text was not segmented into facts and reasoning. It was then segmented based on enumeration symbols, headlines, and similar heuristics.

To ensure high data quality, each such processing step was thoroughly tested via unit tests, and the dataset was manually examined by the authors through a graphical user interface. Incorrectly processed verdicts were then corrected. After filtering verdicts that did not have a guiding principle or a text body, we collected a dataset of 100.018 different verdicts with summarizations. Finally, we divided the datasets into a train (80%), validation (10%), and test set (10%).

3.2 Dataset statistics

We collected all data from German court rulings, such that it can be used in other research such as argument mining, prediction of judgments, etc. The oldest verdict is from 1955, the newest from the end of 2020, but the majority of verdicts were issued between 2010 and 2020. 64% of the verdicts are based on at least one previous court ruling, i.e. they might be based on text which is not directly available to us and some details might be omitted or only referenced to. It is possible to infer information about the most common legal areas by looking at the norms from the normchain. The verdicts most commonly deal with civil cases (BGB, ZPO), administrative jurisdiction (VwGO), financial jurisdiction with mostly income tax law (EStG), or social jurisdiction with mostly asylum law (AsylG, AufenthG).

In comparison to other summarization datasets (see Table 1), our dataset contains documents that are on average 3-4 times longer than the most common summarization datasets CNN, Daily Mail, and NYT. It is comparable with the PubMed and ArXic datasets considering the length of the documents but has the highest compression ratio of all datasets listed. The compression ratio is defined as the document length divided by the summary length. Based on this compression ratio, our dataset is comparable to the BigPatent dataset, another legal dataset for the English language. The datasets' average number of 110 sentences is compressed down to on average 2,9 summary sentences.

To denote the overlap between the summary and the text body novelty (See et al., 2017) quantifies the percentage of n-grams from the guiding principle not found in the text. 13.1% of the guiding principles share all their 1-grams with the text body, whereas 20.4% of the summaries have a 5-grams novelty of 100%. This is an indicator that the summaries have some level of abstraction and are not directly copied from the text body. Abstractive or extractive approaches that select words might thus have an advantage on our dataset.

4 Pre-processing

Given the JSON format described above, we build a pre-processing pipeline specifically tailored to the summarization of German verdicts. Sentences from the guiding principle, facts, and reasoning are first split into tokens on spaces and dots. To reduce the overall complexity, each token is then converted to lower case in order to decrease the number of unique tokens. This is mainly to remove possible mixed-cased words at the beginning of sentences as the meaning of a word is in our case clear given its context. Afterwards norm placeholders in the text are replaced with a generic <*norm*> token, numeric tokens with <*num*> and tokens which were anonymized (via "xx", "..", etc.) by <*anon*>, to infer some semantic meaning for them via word embeddings. Then any tokens within parentheses were removed, as in most cases these only contain references to other legal documents. Afterward, all special characters and any enumeration symbols were removed, as German verdicts contain a lot of structural information. Finally, each token is then mapped to a unique identifier.

After all those processing steps, the whole dataset contains around one million unique tokens, thus we only use tokens that appear at least 100 times in the corpus resulting in a total number of tokens around 50.000. Although this vocabulary size seems rather small, it is comparable to other methods (See et al., 2017) and was necessary to reduce the hardware requirements for the neural networks. All tokens which were not selected are replaced by an <*unk*> token in the pre-processed text. Subword-based tokenization would be one option to removing unknown tokens, but we decided against it as only 2.7% of the text is replaced with <*unk*>. We pre-trained GloVe embeddings (Pennington et al., 2014) on those tokens.

5 Method

The basic building blocks of our models, namely CNNs, RNNs, and the attention mechanisms can be defined by their transformations on a sequence of embeddings $X = x_1, \ldots, x_N; x_i \in \mathbb{R}^d$. For convenience, given a matrix $M \in \mathbb{R}^{n \times m}$ we define $M_{i,:}$ as the i-th row and $M_{:,j}$ as the j-th column vector. The general structure of both model types can be seen in Figure 1 and will be further explained in the respective subsections.

Formally, a 1-dimensional CNN transforms a sequence $X \in \mathbb{R}^{N \times d}$ into the sequence $Y \in \mathbb{R}^{N \times e}$ by[2]

$$\begin{aligned} Y_{i,e_j} &= b_{e_j} \\ &\quad + \left(\sum_{k=0}^{d-1} w_{e_j,k} \star X_{:,k}\right)_i \end{aligned} \tag{1}$$

[2]Definition from pytorch.org

Table 1: Dataset statistics from Cohan et al. (2018) expanded with compression ratio and our dataset. Length denotes the number of tokens. Compression is the percentage of words in the summary compared to the document length. Newsroom and BigPatent statistics taken from Sharma et al. (2019) and XSum from Narayan et al. (2018).

Dataset	#docs	avg. doc length	avg. summary length	compression ratio
CNN	92k	656	43	15.3
Daily Mail	219k	693	52	13.3
NYT	655k	530	38	13.9
PubMed	133k	3016	203	14.9
ArXiv	215k	4938	220	22.4
Newsroom	1212k	751	45	16.7
BigPatent	1341k	3573	117	30.5
XSum	227k	431	23	18.7
Legal	100k	2422	75	32.3

with $\star$ being the cross-correlation operation and b, w learnable weights and hidden state dimensions are d and e.

For the RNN we use a Gated Recurrent Unit (GRU) (Cho et al., 2014) which calculates a new embedding via gates and the previous element in the sequence. We transform a sequence $X \in \mathbb{R}^{N \times d}$ into the sequence $Y \in \mathbb{R}^{N \times e}$ by:

$$\begin{aligned} r_j &= \sigma([W_r X_{t,:}]_j + [U_r Y_{t-1,:}]_j + b_j^r) \\ z_j &= \sigma([W_z X_{t,:}]_j + [U_z Y_{t-1,:}]_j + b_j^z) \\ \hat{h}_j^t &= tanh([W X_{t,:}]_j + [U(r \odot Y_{t-1,:})]_j + b_j^h) \\ Y_{t,j} &= z_j Y_{t-1,:} + (1 - z_j)\hat{h}_j^t \end{aligned} \tag{2}$$

In this case σ is the sigmoid function and $\odot$ the Hadamard product. $W_r, U_r, W_z, U_z, W, U, b$ denote learnable parameters (definition as the PyTorch implementation).

The attention mechanism (Bahdanau et al., 2015) is defined as the weighted sum over the embeddings $x_1, \dots, x_N \in \mathbb{R}^d$ in the following way:

$$a = \sum_{i=1}^{N} \alpha_i x_i \tag{3}$$

α_i is a weighting for every element in the sequence, and can be calculated by taking the softmax over the attention score $e \in \mathbb{R}^N$, which is calculated for the dot-attention via:

$$e_i = s^T x_i \tag{4}$$

In those formulations $s \in \mathbb{R}^d$ is a learnable parameter and the final attention score is $\alpha = softmax(e)$.

5.1 Extractive

Our extractive summarization system is extracting sentences to create a summarization. Given a verdict $V = (f_1, \dots, f_n, r_1, \dots, r_m)$ where f_i denotes a sentence from the facts and r_i denotes a sentences from the reasoning part, we want to find a selection of sentences $S = F \cup R$ with $F \subset \{f_1, \dots, f_n\}$ and $R \subset \{r_1, \dots, r_m\}$ which maximizes the ROUGE score compared to the reference summary. For every sentence s_i we want to find a function which tells us whether to include the sentence in the created summary:

$$f(s_i|V, \theta) = \begin{cases} 1, & \text{if } s_i \in S \\ 0, & \text{else} \end{cases} \tag{5}$$

5.1.1 Gold Labels

The decision on labels for the selection process is not straightforward, as in most cases the guiding principle is not directly copied from the body of the verdict. To overcome this mismatch, it is common to greedily select sentences from the text until the ROUGE score between the guiding principle and the selected sentences does not increase anymore (Nallapati et al., 2017). In our case, we went through the verdict and included a sentence, if it increases the ROUGE-2 F1 score of the current selection.

The only problem with this labeling approach is that the resulting summary is way longer than the guiding principle which has on average 75 tokens. With on average 227 tokens the greedy selection is approximately 3 times longer. To combat this issue, we additionally tested a different selection scheme, where we select one sentence from the body of the verdict which has the highest ROUGE-2 F1 score per sentence in the guiding principle. If

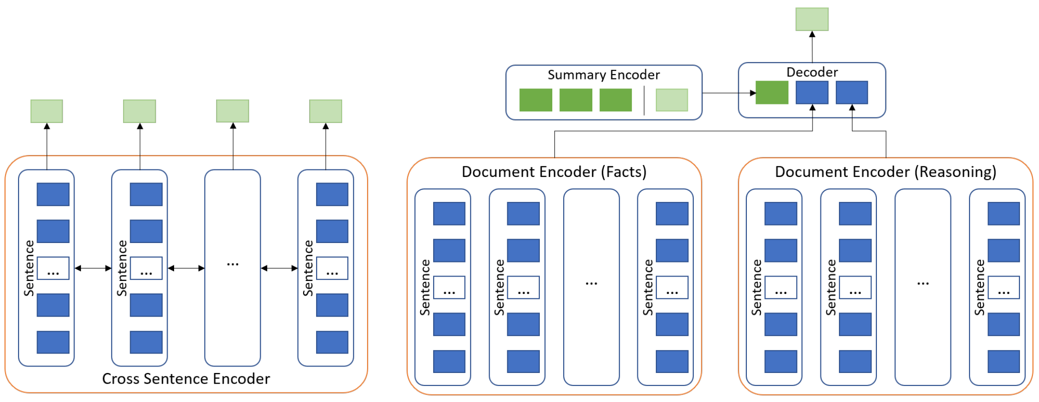

Figure 1: Word embeddings are aggregated to sentence embeddings. For the extractive models (left) information is passed between sentences before classification. In the abstractive case (right) the sentence embeddings are aggregated into document embeddings for facts and reasoning and used in a decoder with the previous words from the summary to predict the current word.

we could not find any overlapping sentences, we used the ROUGE-1 F1 score instead. We then evaluated both labels as an oracle method, to determine what the maximum possible score for each of them would be (see Table 4). There are on average 78 tokens in the one-to-one selection, i.e. lengthwise it is perfectly matching with the reference summary. Surprisingly, the ROUGE scores for the one-to-one selection are higher than the greedy selection. We suspect this is due to the ROUGE-1 criterion in the one-to-one selection. We decided against using ROUGE-1 in the greedy case, as it would further increase the length of the created summary.

5.1.2 Baseline

In addition to the oracle methods, we included other common baselines for the extractive model. One common baseline called lead-3 (Nallapati et al., 2017) simply takes the first 3 sentences of the text as a summary. In our case, we expanded this definition to lead-f-r, which denotes that f sentences are taken from the beginning of the facts and r sentences from the reasoning. Additionally, we also included a random baseline, where we selected 3 random sentences from the whole text. We also tested TextRank (Mihalcea and Tarau, 2004) and LexRank (Erkan and Radev, 2004).

5.1.3 Models

We will now define the models we used for extractive summarization. All following models use the same schema: the token embeddings are processed by multiple Linear layers with ReLU functions in between. Those token hidden states are then used to calculate the sentence embeddings. Linear layers and a Sigmoid function map those to the selection score. The only difference between the models is the way the sentence embeddings are calculated.

We used GloVe (Pennington et al., 2014) to map a word id to a 100-dimensional vector representing its semantics. The sentences are embedded via a CNN, GRU or attention. The CNN model applies multiple 1-dimensional convolutions with ReLU activations in between to the word embeddings. By taking the maximum value over the word embedding dimension, we obtain the sentence embedding. For the RNN model, we pass all the tokens from the sentence into the GRU and then use the embedding of the last word as the sentence embedding. When using attention, the weighted sum is directly used as a sentence embedding.

As the semantics of a sentence are highly dependent on its context, we created cross-sentence layers. The cross-sentence RNN is a bidirectional GRU that passes information around neighboring sentences, the final sentence embedding given by the hidden state of that sentence. The cross-sentence CNN works similarly but uses multiple 1-dimensional Convolutions with ReLUs instead. This way the embedding of a sentence is always dependent on the context of the sentence. We tested the models with and without this cross-sentence information passing.

Each model is trained to optimize the binary cross-entropy loss which is scaled so that the selected sentences gain similar importance to the unimportant sentences. The negative examples are more common, i.e. the training becomes harder without the positive weighting p. The loss for verdict can be formulated in the following way, with V denoting all sentences in the verdict and S denoting the selected summary sentences:

$$L(V,S) = \frac{1}{|V|}(p\sum_{s\in S}\log(f(s|V,\theta)) + \sum_{s\notin S}\log(1-f(s|V,\theta))) \quad (6)$$

We tested all the possible combinations from above. We also fixed the dimensions of each learnable parameter to the embedding size, i.e. during processing each data point has the same dimensionality.

For each training, we choose a random learning rate between 1×10^{-3} and 1×10^{-6} on a log scale, i.e. higher and lower learning rates are equally likely. This was necessary as each model variation needs slightly different learning rates for optimal results and a fixed learning rate would underestimate the performance of a specific model. Each model was trained twice with two different learning rates and we then choose the best model based on the ROUGE-1 F1 score on the validation set. Each model was also tested on the one-to-one and the greedy label selection, i.e. in total, a specific model was trained 4 times. From the two models trained we picked the one with the higher validation set ROUGE-1 performance. Training each model took between 2 and 4 hours.

5.2 Abstractive

The abstractive models produce one word after the other and can be understood as conditional language models which depend on the content of the text and the previously generated words. Given a verdict V we want to approximate the probability distribution over the words w_i in the summary S:

$$P(w_{1,\dots,n}|V) = \prod_{w_i\in S} P(w_i|w_{1,\dots,i-1},V) \quad (7)$$

This can be approximated by a neural network, which is trained to predict the current word given the previous words from the summary and the text as features. For abstractive models we did some minor adjustments to the embeddings: we added one <*end*> token to denote that the model wants to end the current sentence.

5.2.1 Models

The abstractive models use the same building blocks as in the extractive case. Instead of passing information between sentences, they are aggregated to one document embedding. We follow the general encoder-decoder structure. The encoder part of the model is calculating embeddings for the two segments in the body of the court ruling and the previously generated words. The sentences from facts and reasoning are embedded using the attention mechanism from above. To create the document embedding, we apply the RNN model over the sentences where the document is represented by the hidden state of the last sentence. We choose this model combination as it gave us the best extractive performance and thus we can assume that it can adequately model the verdict. The previously generated words are used in a GRU and the hidden state for the last word is forwarded through two linear layers followed by ReLUs to create the wanted embedding.

Given an embedding for the generated words, the facts, and reasoning, the decoder will concatenate their representation and then apply three linear layers with ReLU functions in between. The final layer generates a prediction value for each word which is then transformed to the wanted probability by applying a softmax function. We denote this encoder-decoder model as our *Baseline* model.

As a model variation (*Guided*), the attention can be altered to dynamically change its searching behavior. The dot-attention is changed in the following way with $x_{lin}(.)$ denoting one Linear layer which is swapped depending on if the model is currently encoding the facts or the reasoning section, and $p\in\mathbb{R}^d$ denotes the embedding obtained from the previously generated words i.e. $x_{lin}(.)$ maps from $\mathbb{R}^d$ to $\mathbb{R}^{d_1}$:

$$e_i = x_{lin}(p)^T x_i \quad (8)$$

Inspired by the approaches from Cao et al. (2018); Chen and Bansal (2018) and Bae et al. (2019), we further extend the *Guided* model by using extracted sentences as a template for the encoder part of the network. In this model (*Template*), the encoders for the body of the verdict get the embedding from the extracted sentence instead of the embedding from the previously generated words to dynamically alter the aggregation behaviour based on what should be currently generated. The extracted sentence is aggregated via dot-attention.

During training, we generate the summary word for word, i.e. the model is optimized to predict the current word given the previous words from the gold summary. We optimize the log probability of

Method	Greedy ROUGE-1	Greedy ROUGE-2	Greedy ROUGE-L	One-to-One ROUGE-1	One-to-One ROUGE-2	One-to-One ROUGE-L
Att - x	25.91	8.66	18.07	28.04	10.67	20.21
CNN - x	25.67	8.13	17.90	28.28	10.95	20.64
RNN - x	18.10	2.80	12.08	20.49	4.71	13.72
Att - CNN	23.98	6.62	15.94	18.31	2.98	12.04
Att - RNN	25.07	7.65	17.11	**30.50**	**13.06**	**22.95**
CNN - CNN	18.31	2.98	12.04	29.70	12.30	22.21
CNN - RNN	22.98	5.80	15.15	29.63	12.18	22.02
RNN - CNN	18.20	2.87	12.14	21.17	5.16	14.18
RNN - RNN	22.13	5.96	15.42*	22.05	5.88	15.30*

Table 2: Test performance levels for extractive methods. The methods are written as *sentence - cross-sentence* encoder. We pick the top-3 sentences for each extractive method. ROUGE-L score marked with * is not exact as we needed to remove some verdicts due to recursion errors induced by long generated summaries.

the given word and sum the loss over the whole summary sequence:

$$l(w_i|V,\theta) = \log(f_\theta(w_i)) \tag{9}$$

$$L(w_1,\ldots,w_n) = \sum_{w_i \in S} l(w_i|V,\theta) \tag{10}$$

We fixed the learning rate to 5×10^{-3} for the training as each epoch takes multiple hours. As evaluation on an epoch level would be very time-consuming, we instead evaluated the model every 1000 verdicts on 100 verdicts from the validation set. 10 verdicts are batched together for optimization. After the 80th such iteration, we decrease the learning rate by a factor of 0.95 for 20 iterations. We used early stopping with a patience of 30 on the validation loss and training each model took 4-5 days.

For testing, the summarization model is allowed to produce at most 150 tokens or 3 sentences denoted by the <*end*> token. Each summary is then generated word for word, where the model gets the previously generated words as input as well as the text of the verdict. We used the one-to-one extractive labels for *Template*. We use beam search (Rush et al., 2015) with a width of 5 summaries, i.e. in each step, the top-5 summaries based on the joint probability are used for further generation. This heuristic search is stopped if one sentence reaches the ending conditions from above.

As a baseline to the abstractive approaches, we used the Pointer-Generator network from See et al. (2017). We used the same vocabulary size as with the other models, but otherwise, the hyperparameters were not changed.

6 Performance Analysis

To evaluate a given system, we use the ROUGE metric (Lin, 2004) which measures the percentage of n-gram overlap between a gold label summarization $V_{sum} = (w_1,\ldots,w_n)$ and a created text $C = (c_1,\ldots,c_m)$ with $w_i : i \in \{1,\ldots,n\}$ and $c_i : i \in \{1,\ldots,m\}$ being the words in the respective texts [3].

In Table 2 we denote the test performance for the extractive methods. In most cases, the model with the greedy target selection performs worse than the one-to-one selection. Models trained on the one-to-one labels might gain a better separation between important and irrelevant sentences and thus we observe a higher summarization score. In our case, a GRU is less desirable for sentence encoding. The models with RNNs as sentence encoders have a maximum ROUGE-1 score of 22.13 percent. As seen in Table 2 our CNN and attention models have a higher level of performance by a wide margin compared to the approaches which use RNNs as sentence encoders. By introducing a cross-sentence layer, the overall performance does increase slightly.

The performance for the abstractive models (Table 3) is worse compared to the extractive models and the baselines. Our current abstractive approaches are not able to adequately model the summarizations. There is a performance increase for the Pointer-Generator model (See et al., 2017), but its summaries are also rather poorly generated. The common problem for all tested abstractive models was that they tend to repeat phrases, thus the quality

[3]Evaluated with the following implementation: `https://github.com/pltrdy/rouge`

Table 3: Test performance for the abstractive models. Baseline and Guided method have converged to a similar language model, but they produce slightly different probability for the words.

Method	ROUGE-1	ROUGE-2	ROUGE-L
Baseline	12.37	0.77	9.96
Guided	12.37	0.77	9.96
Template	10.86	0.65	8.32

Table 4: Test performance for the best models, compared to the baselines. For methods marked with *, we had to exclude some verdicts with long summaries, due to recursion depth of the ROUGE-L calculation.

Method	ROUGE-1	ROUGE-2	ROUGE-L
oracle - One-to-One*	55.12	40.39	51.38
oracle - Greedy*	37.90	26.10	35.16
lead-3-0	11.80	1.52	7.75
lead-0-3	18.44	2.88	12.38
lead-3-3	19.69	3.18	12.42
random	19.95	3.90	13.12
TextRank (Mihalcea and Tarau, 2004)	22.69	5.93	14.95
LexRank (Erkan and Radev, 2004)	24.98	6.87	16.34
Pointer (See et al., 2017)	20.80	5.27	15.98
Ext: Att - RNN	**30.50**	**13.06**	**22.95**
Abs: Baseline	12.37	0.77	9.96

of the summary degrades, which is also reported by other researchers (See et al., 2017). Further research is necessary to investigate whether those low-performance levels are also produced by other abstractive summarization approaches.

Table 4 shows the test performance for all the best models, together with the baselines. Our extractive approach beat all baselines, but we still have room for improvement as seen by the oracle performances. Furthermore, as we can report similar oracle performances to other papers (Xu et al., 2020; Liu and Lapata, 2019), our hypothesis of using summarization techniques for the generation of guiding principles is strengthened.

When comparing the performance numbers with those reported at `nlpprogress.com`, we see the long historic evolution for summarization methods. The legal domain and especially the German language are under-researched in the summarization domain. A lot of work, such as creating appropriate language models is necessary to apply those methods to the German language. With certainty, we can say that there is a necessity to evaluate summarization models on diverse datasets.

7 Conclusion & Future Work

In this paper, we introduced a new dataset for the summarization of German court rulings. We created multiple models specifically tailored to the German legal domain and tested strong baseline models which we want to improve in the future.

Nonetheless, there are still extensions and shortcomings which we want to address in future work. The vocabulary size needs to increase as German is a more morphologically rich language than English. It might be necessary to find more efficient ways to represent and generate compound nouns, the different verb forms, etc. as they drastically increase the size of the necessary vocabulary. Consequently, the model complexity needs to be increased. This does not only include ways to better model the German language but also taking the structure of the verdicts more into account. Court rulings have a more sophisticated structure which we did not entirely utilize in this research paper, i.e. each of the segments is again subdivided into other sub-segments and so on. Embracing this more hierarchical view could make access to information easier and improve the understanding of the source document.

In this research, we laid down the necessary groundwork for further research. In the future, we want to do more qualitative analysis, by letting legal experts evaluate our summarization models. Summarization is far from being solved and in our opinion, it is important to introduce different texts and domains to summarization research.

References

Sanghwan Bae, Taeuk Kim, Jihoon Kim, and Sang-goo Lee. 2019. Summary level training of sentence rewriting for abstractive summarization. In *Proceedings of the 2nd Workshop on New Frontiers in Summarization*, pages 10–20, Hong Kong, China. Association for Computational Linguistics.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *International Conference on Learning Representations*.

Ziqiang Cao, Wenjie Li, Sujian Li, and Furu Wei. 2018. Retrieve, rerank and rewrite: Soft template based neural summarization. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 152–161, Melbourne, Australia. Association for Computational Linguistics.

Asli Celikyilmaz, Antoine Bosselut, Xiaodong He, and Yejin Choi. 2018. Deep communicating agents for abstractive summarization. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1662–1675, New Orleans, Louisiana. Association for Computational Linguistics.

Yen-Chun Chen and Mohit Bansal. 2018. Fast abstractive summarization with reinforce-selected sentence rewriting. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 675–686, Melbourne, Australia. Association for Computational Linguistics.

Jianpeng Cheng and Mirella Lapata. 2016. Neural summarization by extracting sentences and words. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 484–494, Berlin, Germany. Association for Computational Linguistics.

Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, Doha, Qatar. Association for Computational Linguistics.

Sumit Chopra, Michael Auli, and Alexander M. Rush. 2016. Abstractive sentence summarization with attentive recurrent neural networks. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 93–98, San Diego, California. Association for Computational Linguistics.

Arman Cohan, Franck Dernoncourt, Doo Soon Kim, Trung Bui, Seokhwan Kim, Walter Chang, and Nazli Goharian. 2018. A discourse-aware attention model for abstractive summarization of long documents. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 615–621, New Orleans, Louisiana. Association for Computational Linguistics.

Li Dong, Nan Yang, Wenhui Wang, Furu Wei, Xiaodong Liu, Yu Wang, Jianfeng Gao, Ming Zhou, and Hsiao-Wuen Hon. 2019. Unified language model pre-training for natural language understanding and generation. In *33rd Conference on Neural Information Processing Systems (NeurIPS 2019)*.

Günes Erkan and Dragomir R Radev. 2004. Lexrank: Graph-based lexical centrality as salience in text summarization. *Journal of Artificial Intelligence Research - JAIR*, 22:457–479.

Sebastian Gehrmann, Yuntian Deng, and Alexander Rush. 2018. Bottom-up abstractive summarization. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4098–4109, Brussels, Belgium. Association for Computational Linguistics.

Ingo Glaser, Sebastian Moser, and Florian Matthes. 2021. Sentence boundary detection in german legal documents. In *Proceedings of the 13th International Conference on Agents and Artificial Intelligence - Volume 2: ICAART*, pages 812–821. INSTICC, SciTePress.

Aishwarya Jadhav and Vaibhav Rajan. 2018. Extractive summarization with SWAP-NET: Sentences and words from alternating pointer networks. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 142–151, Melbourne, Australia. Association for Computational Linguistics.

Anastassia Kornilova and Vladimir Eidelman. 2019. BillSum: A corpus for automatic summarization of US legislation. In *Proceedings of the 2nd Workshop on New Frontiers in Summarization*, pages 48–56, Hong Kong, China. Association for Computational Linguistics.

Wojciech Kryściński, Romain Paulus, Caiming Xiong, and Richard Socher. 2018. Improving abstraction in text summarization. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1808–1817, Brussels, Belgium. Association for Computational Linguistics.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational*

Linguistics, pages 7871–7880, Online. Association for Computational Linguistics.

Wei Li, Xinyan Xiao, Yajuan Lyu, and Yuanzhuo Wang. 2018. Improving neural abstractive document summarization with structural regularization. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4078–4087, Brussels, Belgium. Association for Computational Linguistics.

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain. Association for Computational Linguistics.

Yang Liu and Mirella Lapata. 2019. Text summarization with pretrained encoders. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3730–3740, Hong Kong, China. Association for Computational Linguistics.

Rada Mihalcea and Paul Tarau. 2004. TextRank: Bringing order into text. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*, pages 404–411, Barcelona, Spain. Association for Computational Linguistics.

Ramesh Nallapati, Feifei Zhai, and Bowen Zhou. 2017. Summarunner: A recurrent neural network based sequence model for extractive summarization of documents. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, AAAI'17, page 3075–3081. AAAI Press.

Ramesh Nallapati, Bowen Zhou, Cicero dos Santos, Çağlar GuÌ‡lçehre, and Bing Xiang. 2016. Abstractive text summarization using sequence-to-sequence RNNs and beyond. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*, pages 280–290, Berlin, Germany. Association for Computational Linguistics.

Shashi Narayan, Shay B. Cohen, and Mirella Lapata. 2018. Don't give me the details, just the summary! topic-aware convolutional neural networks for extreme summarization. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1797–1807, Brussels, Belgium. Association for Computational Linguistics.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Weizhen Qi, Yu Yan, Yeyun Gong, Dayiheng Liu, Nan Duan, Jiusheng Chen, Ruofei Zhang, and Ming Zhou. 2020. ProphetNet: Predicting future n-gram for sequence-to-SequencePre-training. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 2401–2410, Online. Association for Computational Linguistics.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. arXiv preprint.

Alexander M. Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 379–389, Lisbon, Portugal. Association for Computational Linguistics.

Abigail See, Peter J. Liu, and Christopher D. Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1073–1083, Vancouver, Canada. Association for Computational Linguistics.

Eva Sharma, Chen Li, and Lu Wang. 2019. BIGPATENT: A large-scale dataset for abstractive and coherent summarization. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2204–2213, Florence, Italy. Association for Computational Linguistics.

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015. Pointer networks. In *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc.

Jiacheng Xu, Zhe Gan, Yu Cheng, and Jingjing Liu. 2020. Discourse-aware neural extractive text summarization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5021–5031, Online. Association for Computational Linguistics.

Jingqing Zhang, Yao Zhao, Mohammad Saleh, and Peter Liu. 2020. PEGASUS: Pre-training with extracted gap-sentences for abstractive summarization. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 11328–11339. PMLR.

Xingxing Zhang, Furu Wei, and Ming Zhou. 2019. HIBERT: Document level pre-training of hierarchical bidirectional transformers for document summarization. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5059–5069, Florence, Italy. Association for Computational Linguistics.

Qingyu Zhou, Furu Wei, and Ming Zhou. 2020. At which level should we extract? an empirical analysis on extractive document summarization. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 5617–5628, Barcelona, Spain (Online). International Committee on Computational Linguistics.