Proceedings of the 55th Hawaii International Conference on System Sciences | 2022

Identification of Decision Rules from Legislative Documents Using Machine Learning and Natural Language Processing

Maximilian Michel
TU Vienna
max.michel@km-h.at

Djordje Djurica
WU Vienna
djordje.djurica@wu.ac.at

Jan Mendling
Humboldt University Berlin
jan.mendling@hu-berlin.de

Abstract

Decision logic extraction from natural language texts can be a tedious, labor-intensive task. This is especially true for legislative texts, since they do not always follow usual speech and writing patterns. This paper explores the possibility of using machine learning and natural language processing approaches to identify decision rules within legislative documents, and ultimately provides the possibility of building an extraction algorithm on top of the solution to extract and visualize decision logic automatically. Such a novel method for decision rules identification bears the potential to reduce human labor, minimize mistakes, and lessen context dependency. To accomplish this, we use pre-trained word vectorization in conjunction with a complex multi-layer convolutional neural network (CNN). The relevant data used in this project was generated from the Austrian income tax code and labeled by hand. A quantitative evaluation shows that our approach can be trained on as little as a single code of law and still obtain significant accuracy.

1. Introduction

When implementing the automated rule and compliance checking into a digitally aided workflow, we face the challenge of an often prohibitive, extensive specification effort. The legislative documents that comprise most legal systems express rules implicitly in the form of natural language. Building on human expertise is still the prominent solution to bridge the gap between natural language and a machine comprehensible version of its content. The translation task is knowledge-intensive, slow, and often comes with high costs. Therefore, the extraction of rules and other semantic content from legal sources has been a prevalent topic in natural language processing.

Different approaches to extract decision logic have been presented in both information systems and computer science. The approach proposed by Rozinat and van der Aalst [1] present first efforts in mining decisions from transaction logs or audit trails through the use of machine learning techniques. More recently, work by van der Aa et al. [2] focused on extracting decision logic and visualizing it as declarative process models from the natural language, while Arco et al. [3], and Etikala et al. [4] focused on decision rule extraction and the decision table and decision requirements diagram generation. Research on rule-based approaches often suffers from a lack of annotated source data, and the inherent complexity of extracting decision logic from natural texts [5, 6, 7, 8]. While these approaches yield promising results individually, they are closely tied to the specifics of use cases. In particular, when using gazetteers [6], significant work might be needed to adapt them to different content and context.

In this paper, we address the research problem of devising an easily adaptable and scalable identification of decision rules in legislative texts. To accomplish this, we build on prior research by Kim on sentence classification using convolution neural networks [9]. In our approach, we first classified sentences from the Austrian Income Tax Law based on their deontic expressions: decisions, obligations, permission, and prohibition. Then, we implemented a complex multi-layer convolutional neural network using a pre-trained word vectorization. This resulted in a scalable method, on top of which rule-based extraction algorithms can easily be built. Our research suggests that training data with a number of examples in the lower thousands are sufficient to train the neural network, provided care is taken when designing the model [10]. Additionally, advances in word embedding have made it possible to create vectorized representations of words that keep much of their semantic information [11] allowing for finer-grained classification on the level of the sentence.

This paper is structured as follows. Section 2 gives a brief overview of prominent approaches to information extraction from natural language, especially legislative texts. Section 3 describes in detail the steps necessary to collect and preprocess the data and create the neural

URI: https://hdl.handle.net/10125/80097
978-0-9981331-5-7
(CC BY-NC-ND 4.0)

network. Section 4 presents the results. The Discussion in Section 5 compares the baseline approach, lists the project's limitations, and provides insight into the resulting implications. Section 6 serves as a conclusion and discusses directions for future research.

2. Theoretical Background

2.1. Rule Extraction Tasks in a Legal Context

In order to analyze automated rule extraction from legislative language sources, the machine-readable representation of rules in the text has to be discussed first. Regarding rule extraction, the focus lies on the semantic content of individual clauses in a legal document. Clauses can be categorized into two general groups: definitional clauses and normative clauses [12]. Definitional clauses provide definitions regarding terms and concepts used throughout the source, while normative clauses comprise deontic modalities, a linguistic concept indicating how something ought to be. This includes obligations, permissions, and prohibitions [13]. For the deontic concepts, annotations can be added stating the type of deontic expression and defining the thematic roles agent and theme. The role *agent* marks the active entity, *theme* the entity that is acted on. These roles are useful due to the subject and object of a sentence often switching when both active and passive structure is used throughout the document [6].

2.2. Related Work

Prior work can be largely grouped into two types of solutions: A rule-based approach and a machine learning approach.

2.2.1. Rule-Based Approach Rule-based methods have their creators manually construct a framework of rules and routines for text processing. When using such methods, the document is first run through a parser that annotates the grammatical structure of each sentence. Then, the output is evaluated, extracting rules based on syntactic and linguistic patterns. Therefore, it is necessary to understand how exactly information is encoded in the relevant document. For all-natural language texts, attention must be paid to grammar, vocabulary, and context. However, comprehension of formal structure, e.g., bullet points and lists, can be relevant in legal documents.

An example of such an approach is work by Wyner and Peters [6] which presents "a linguistically-oriented, rule-based approach" to rule extraction structured around the use case of compliance issues in highly regulated industries such as the banking or health service sector. A 1700-word document generated from US Food and Drug Administration regulations regarding blood banks is annotated with the help of a linguistic parser (Stanford Parser[1]). The resulting grammatical annotations are then combined with gazetteer lists and regular expressions to identify and mark relevant elements like agent and theme, deontic modals, main verbs, exception clauses, and conditional sentences. A comparison to a previously created gold standard for this document reveals that this method achieves promising results, especially for more straightforward concepts like deontic modalities and list structures. While these achieve the ideal result of 100% in both precision and recall, the more complex patterns like exception phrases and by-phrase agents can match this only in precision with recall lacking behind with rates often under 50%.

A different model using similar components is presented in [5]. In contrast to [6] the evaluation is not a linear process but conducts two operations in separate branches, only crosschecking the results at the end. The document is again split into sentences in the first branch, which are consequently parsed with the Stanford Parser. Then, however, each phrase is further divided into terms. In this case, "terms" do not represent single words, but whole sentence components are isolated by their position in a sentence's grammatical structure. Each term is annotated with a deontic tag based on a thesaurus created with the help of the large lexical database WordNet[2]. Resulting in sentences split into deontically tagged terms which can be arranged to form a logical representation of the described rule. The second branch creates a phrase structure representation of a sentence using a combinatory categorical grammar parser. Finally, the logical relationships between terms created by the first branch are checked against the logical representation created by the second.

The paper by Etikala et al. [4], presents a text mining technique that is capable of automatic extraction of decisions and their dependencies from natural language text and building the decision requirements diagram (DRD). This approach labelled as Text2Dec uses open-source tool kits such as Stanford's core NLP [3], NLTK [4], neuralcoref [5], and SpaCy [6] libraries to build the NLP pipeline which enables decision logic extraction. This approach was evaluated using a real-life use case of *Employee Health Assessment*. The results showed that Text2Dec could generate DRDs that correspond to

[1]https://nlp.stanford.edu/software/lex-parser.shtml
[2]https://wordnet.princeton.edu/
[3]https://nlp.stanford.edu/software/
[4]https://www.nltk.org/
[5]https://github.com/huggingface/neuralcoref
[6]https://spacy.io/

the manually developed ones in terms of structure and semantics.

Another recent effort regarding automatic discovery and generation of decision rules and decision tables from natural language descriptions is the work by Arco et al. [3]. Their approach consists of discourse and semantic analysis, syntactic analysis, and decision table generation. To accomplish this, the authors make use of Stanford Parser. The evaluation shows that this framework can identify and extract decision rules from decision descriptions in English with high accuracy for the antecedent-consequent pairs and their components. Some limitations regarding the syntax of the approach compared to expert modelers' syntax, resulting in occasional decision table extraction that does not correspond to the experts' expectations.

All the examples presented above show promising results, but precision and recall rates often fluctuate between different categories of deontic concepts. Additionally, the tests were performed on small, highly structured, and manually preprocessed data. Therefore, reproducibility on other documents is not guaranteed. The exception to this is the paper by Parkash et al. [14], which was tested on multiple data sets. However, even this study points that the presented tool (BRMiner) performs well, as long as the sentences do not consist of too many dependent clauses. They further state that their approach can be used to further train and utilize supervised learning-based models.

2.2.2. Machine Learning Approach With supervised machine learning, the exact rules for obtaining a result do not have to be clear from the beginning. On the contrary, machine learning creates those rules by finding patterns in the provided data. To train such a model, data and the result to be predicted have to be aggregated in a training set. The model "learns" to generate the wanted output based on the respective input datum. When presented with new data, it will generate predictions that fit the patterns encountered in the training data. A model trained on a more extensive data set will have more chances to recognize relevant patterns. Therefore, large data sets are generally required for model training. Another requirement is the data being in a format the machine learning algorithm can work with. In general, a machine learning model operates on a numeric representation of the task at hand.

A simple solution for natural language is the bag-of-words (BOW) method [15]. For each document, only the absolute number of occurrences or the frequency of a word in the document relative to the whole corpus is kept. Disregarding the context and position a word is in this drastically simplifies the content of a text, but in turn, enables classification based on relative term frequencies.

In Mikolov et al. [11] and successively in Bojanowski et al. [16] a more complex alternative to the simple BOW representation is presented. Mikolov et al. [11] demonstrate that using their Skip-gram method, a word can be represented as a multidimensional vector and keep its context information. Thereby, even complex relationships between terms can be preserved when translating them into a numeric format. Bojanowski et al. [16] improve on this concept by including subword information in their vectorization model. This enables the resulting vector to include context information based on the position of a word in a sentence or document and based on the position of structurally similar terms. Conducted tests show that this improves results in word similarity tests and enables the model to vectorize words that are not included in the training data as long as similar terms are included in sufficient quantity.

The paper by Elwany et al. [17] presents the use of BERT [18] on legal documents. This paper aimed to assess the performance of a model for classification tasks based only on a pre-trained BERT language model. For BERT fine-tuning, the authors used a proprietary corpus consisting of hundreds of thousands of legal agreements, while as a classification data set, they used a proprietary data set consisting of a few thousand legal agreements. This paper showed that the pre-trained BERT model significantly improves the classification tasks in the legal domain.

For even finer grouping, in [9] a convolutional neural network (CNN) is used to classify single sentences. An input vector is connected to its corresponding output vector through a net of consecutive layers of nodes in a neural network. These nodes also called "neurons," take a weighted vector and run it through a mathematical function to calculate the output. The model "learns" patterns in the data by iteratively running through the provided input vectors and adjusting the weights for each node to fit the desired output vector provided in the training set. A CNN follows this principle but differs from a simple "feed-forward" neural network in that it does not connect all its input neurons to every node on the next layer. The nodes of a layer are only connected to individual areas in the input data. When used for image recognition, the model does not look at the whole picture at once but singles out smaller areas to find patterns first. These are accumulated over the subsequent layers to form complex filters, enabling the model to make sophisticated classifications.

Kim [9] conducted multiple tests comparing four variants of a convolutional neural network to other state-of-the-art sentence classification methods. Of the four models, three are trained on word vectors created based on the concepts mentioned above. One, serving as a baseline, is trained on randomly initialized vectors. All models are tested on seven standard benchmark data sets, including tasks such as multi-class classification of Questions and sentiment prediction of movie and customer reviews. Results show that for every data set, the model trained on randomly initialized word vectors is outperformed by the ones operating on pre-trained vector representations. Furthermore, the CNNs posted the highest result for four of the seven data sets.

3. Research Method

Following the previously presented approaches rooted in machine learning, we implemented a CNN to label individual sentences based on their deontic content. After data is aggregated and labeled, the network was trained and then tested to determine if the proposed model can reliably reproduce the classification when applied to new data. In the following subsections, we describe how our model was constructed and evaluated. An overview of the approach can be seen in Figure 1.

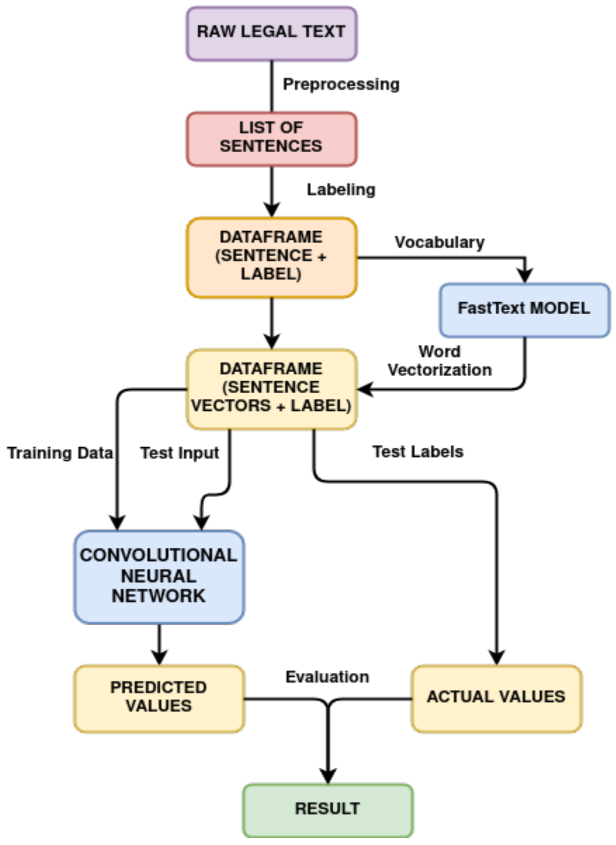

Figure 1. Workflow of the Implemented Approach

3.1. Classification Labels

Before aggregating and labeling relevant data, the different categories of sentences are classified and defined. Primarily, the focus lies on identifying normative clauses and definitional clauses. Definitions clauses serve as a de facto catch-all class, which includes sentences that are not defining the behavior of taxpayers and tax authorities. Normative clauses are to be further divided in three subdivisions: *obligation*, *prohibition* and *permission*. The three subdivisions are based on deontic logic concepts [19] obligation (*OB(a)*), prohibition ($OB(\neg a)$), permission ($\neg OB(a)$), omission ($\neg OB(\neg a)$), and option (($\neg OB(a) \& \neg OB(\neg a)$)). The last three have due to their similarity all been grouped under the label *permission*. This leaves sentences to each be labeled as one of four distinct classes: *definition*, *obligation*, *prohibition* and *permission*. To be compatible with the numeric machine learning model each label is represented by a four dimensional vector with a single binary digit indicating the occurrence of an expression (e.g. Def $\rightarrow$ [1,0,0,0], Obl $\rightarrow$ [0,1,0,0]).

3.2. Data Preparation

3.2.1. Data Source and Preprocessing The Data Set used to train the machine learning model in this project is constructed from the publicly available Austrian Income Tax Code (Einkommensteuregesetz 1988[7]). The choice of resource is three-fold. Firstly, tax law can be considered as relevant for compliance considerations both to an enterprise and the legislators. Secondly, we assume that tax law has less room for interpretation and a more orderly structure than, for example, civil law. Finally, the Austrian income tax code is a well-maintained and relatively new code of law, reducing outdated terms and expressions.

After choosing the text, we proceeded with the data preparation in a semi-automated manner. We separated the text of the law into single sentences using a programming script. However, complex listings that finish the start of a sentence with a different ending for each sub-letter had to be manually identified, and after identification, each listing was split up into individual grammatically and semantically correct sentences. Therefore, the output of the utilized script was a list of individual sentences, which were subsequently manually labeled. Although tedious, this process provided the opportunity to check the automatically aggregated sentences for grammatical and structural coherence. Working through the rows showed that the

[7]https://ris.bka.gv.at/eli/bgbl/1988/400/P0/NOR40205159

preprocessing scripts work as intended and that only a handful of sentences must be removed. The remaining phrases were stripped of unnecessary components such as punctuation, numbers, and unnecessary white space. Next, all abbreviations and symbols were converted to their worded counterparts, and everything was converted to lowercase. Lastly, sentences with a length of more than 64 words were discarded to keep the size of a single input datum reasonably small, preventing a huge input layer in the machine learning model.

In order to make the most of the limited amount of labeled data available for this project, simpler statistical word representation methods were disregarded in favor of a neural network-based approach, in this case, the FastText[8] library. FastText translates every word into an n-dimensional vector that makes it uniquely identifiable and keeps much of its semantic meaning. The framework achieves this by using both learned word embeddings from a continuous skip-gram model as introduced by Mikolov et al. [11] and sub-word information in the form of n-grams introduced by Bojanovski et al. [16]. This has the advantage of providing both context and sub-word information in one vector, enabling words to be represented correctly even if they rarely occur or never in a document corpus, as long as words consisting of similar n-grams are present in sufficient quantity. This is especially useful or even essential when working with German text due to German nouns often being concatenations of multiple ones that can otherwise stand alone. In legislation, this is often used to create precise technical terms. These would otherwise be difficult to vectorize, as their occurrences are sparse or non-existent throughout the training data, preventing a skip-gram only model from learning them. Using this approach, a dictionary comprising all unique words in the data set is generated in applying the concepts mentioned above. Each word in the dictionary is then converted to its vector using FastText's publicly available pre-trained vectorization model for the German language[9]. With the thereby obtained look-up table of words and their 300-dimensional vector representation, each sentence is translated to a list of vectors.

3.2.2. Final Structure The amount of sentences in the classification categories was not evenly distributed. Therefore, to have as many sentences as possible for the test set, we decided to randomly split the data set into a training and a test set in a ratio of roughly 9:1 instead of the usual 8:2. We acknowledge that this proportion can result in slightly different results than using the usual 8:2 proportion. As it can be observed in Table 1, the distribution of classes in the entire data set is carried over both in the training and the test set.

	Full Data	Training Set	Test Set
Definition	1190	1083	107
	42%	42%	43%
Obligation	1274	1163	111
	45%	45%	44%
Permission	265	240	25
	9%	9%	10%
prohibitions	104	97	7
	4%	4%	3%

Table 1. Total and relative class label distribution.

3.3. Machine Learning Model

3.3.1. Vectorization Convolutional neural networks have first been introduced to be used in image recognition in the 1990s [20]. On smaller data sets with limited classification levels like the MNIST digit-recognition set, machine learning algorithms applying these concepts are achieving almost human performance levels with error rates lower than 0.3% [21]. Even on large sets with thousands of classes, CNNs show respectable results with error rates of around 15% [22].

For this project, in trying to extend the advantages and achievements of CNNs to natural language classification, sentences are essentially treated as images. The vectorization described in section 3.3.1 converts each word into 300 numeric values between -1 and 1. To format a sentence like an image, each word vector of 300 values is written in a new row creating a matrix with 300 columns, very similar to a 300 by *n* pixel image, *n* being the number of words in the sentence. With the data formatted like this, the same methods perfected for image classification can be used on sentences without any changes to the architecture. Similarly, as described by Kim, [9] a convolutional neural network is used to classify single sentences. The constructed network is of increased complexity to mitigate the disadvantage of having only a small amount of training samples, as suggested in Lopez and Kalita [10]. Therefore, rather than having a single convolutional, pooling, and fully connected layer each, this network consists of multiple convolutional and pooling layers feeding into a fully connected sequential neural net with five hidden layers. A visualization is provided in Figure 2.

3.3.2. Input The network takes as input a matrix with 300 columns and 64 rows. To standardize the input and cope with different sentence lengths, each input is padded to the required length by adding rows of zeros to the end

[8]www.fasttext.cc
[9]https://fasttext.cc/docs/en/crawl-vectors.html

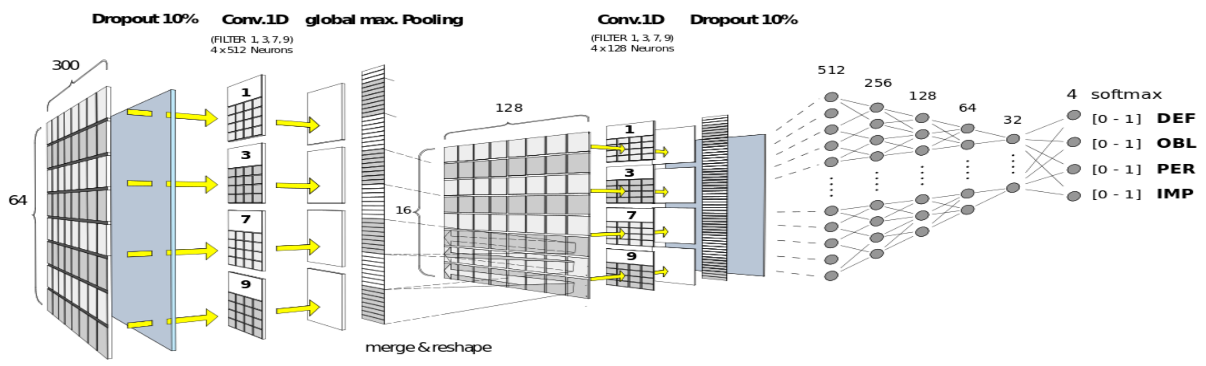

Figure 2. Structure of the Convolutional Neural Network

until a length of 64 is reached. A matrix is introduced since most sentences are much shorter than the maximum length bias of relevant information being mostly situated at the beginning of each sentence. Later layers address this problem, for it could lead to the model identifying patterns that occur later in a long sentence worse or not at all.

3.3.3. Dropout When a complex model is trained on a small data set, it is prone to overfitting, especially when training the model in repeated epochs on the same data. One simple solution without the need for an increase in training data is "dropout," a concept introduced by Hinton et al. in [23]. A dropout layer randomly discards a fixed percentage of the values in each input datum before handing it over to the next layer. The following layers never get to see the whole datum, preventing memorizing all individual training examples and forcing them to learn the meaning of a single value detached from other units around it. This essentially achieves the same result as model averaging, a process in which many models are trained, and their results on the test data are averaged, at a fraction of the computational cost [23].

The here described model includes two dropout layers, one in the beginning before feeding into the first convolutional layers and a second in front of the fully connected sequential network at the end (see fig. 2). Instead of a dropout percentage of 50% as described in [23], each of the layers discards only 10% of its input units. The comparably low dropout rate results from several tests showing that high and low rates prevented overfitting equally well, but higher rates also reduced the overall accuracy in both test and training sets.

3.3.4. Convolutional Layers Each of the first four parallel convolution layers has the same number of 512 filters, but the window size varies. The first layer has the filter window set to one row (1-word vector or 300 units) of the input matrix. For the remaining three layers, the stride length stays the same. However, the window size increases to 3, 7, and then 9 rows, or respectively 900, 2100, and 2700 units. The smaller window sizes are supposed to enable the model to learn features to single words and their positions, while the larger ones should create longer sub-sentence features. All convolution layers use the Rectified Linear Unit (RELU) [24] as activation function. The RELU function $f(x) = max(0, x)$ is non-saturating, mathematically defined as $|\lim_{z\to\infty} f(z)| = +\infty$, meaning it does not confine its input into a range of real numbers, as is done for example by the logarithmic function ($f(t) = \frac{1}{1+e^{-t}}$ aka. Sigmoid). Non-saturating functions allow for faster training due to not flattening the gradient of its output at the borders [22]. For the second set of convolution layers, only the filters and size of the filter windows are adapted to fit the differently structured input received from the merging and reshaping operations. The details of which are explained in section 3.3.5. The task of the second set is to combine and reduce the many simple features created in the first set into less but more complex features.

3.3.5. Pooling The resulting filter maps obtained from each convolution layer are then fed into a global max-pooling function. For the convolution layers in the first set, this means that all 2048 (4 x 512) filter maps, previously of different sizes due to varying filter window sizes, are reduced to a single value each. The pooling operation is again a computationally inexpensive method to address multiple problems mentioned above and helps improve the model's overall effectiveness. The first general benefit of applying a pooling layer after a convolution is to select out positions where the filter matches best. Pooling reduces how well the pattern is represented in different areas of a sentence to the single value of the one location it fits best. It significantly decreases the volume of data passed on to later layers, separating the important activation signals from the ones less so [25]. Secondly, concentration on relevant values has increased a model's ability to generalize [23] mitigating the risk of overfitting. Lastly, pooling helps to rule out bias introduced into the model by varying sentence length in the input data [9]. Due to padding each input datum with trailing zeros to achieve a uniform length, the information density is highest at the beginning of each datum. Without mitigation, this would lead to a part of the network to mostly encounter zeros, hindering learning and possibly leading to patterns occurring only at the end of long sentences not to be recognized. With pooling after the first layer, this effect is contained to the beginning of the network, assuring that later layers will receive information equally proportioned to all nodes.

The model includes a pooling layer after each convolution layer. This transforms the output of each layer, 512 matrices of varying sizes, into four vectors of 512 values each. Before they are handed forward to the next layer, the four vectors are merged and restructured to a 128 by 16 matrix, again creating an image-like format(see merge & reshape in fig. 2). After the second set of pooling layers, the output, four-vectors with 128 values each, are only merged without reshaping to a matrix due to the following fully connected layers not requiring any special input formatting.

3.3.6. Sequential Neural Net The last section of the model comprises a five-layer deep, fully connected feed-forward neural net and a four-node softmax output layer. Its task is to assign each input datum to one of the four classes derived in section 3.1 based on the features learned in the convolution layers. In each subsequent layer, the number of nodes is halved, reducing the number of features from 512 to 32 over 4 layers. Fully connected means that every output of every node in a layer is calculated by taking the weighted sum of all outputs of the previous layer and applying an activation function to it. In this model again the rectified linear unit is used:

$$output = relu(\sum_{i=1}^{n} x_i \times w_i)$$

Where x_i is the output value of node i of the previous layer, n is the number of nodes in the previous layer, w_i is the weight this value is multiplied with and $relu()$ is the aforementioned rectified linear unit activation function $f(x) = max(0, x)$. Therefore, the result of one layer is a vector of values between 0 and ∞ with a length corresponding to the number of nodes in the layer. However, the output layer functions the same way as its predecessors, using the Softmax activation function. Softmax returns a value between zero and one for each node, with the additional property that the sum of all nodes always equals one. The output can be read as a probabilistic estimate of the classification label ([0.05, 0.8, 0.05, 0.1] $\rightarrow$ [0,1,0,0]).

Epochs:	48
Training Acc.:	99%
Definition Acc.:	92.5%
Obligation Acc.:	97.3%
Test Acc.:	93.6%
Permission Acc.:	92%
Prohibition Acc.:	58.1%
F1:	87.4%
Precision:	90.1%
Recall:	84.7%

Table 2. Summary of the model's results.

3.3.7. Training In a first step, a "loss" function measuring how close the model got to recreating the label provided with the input datum has to be defined. For this project, the Huber loss [26] function was applied. It presents a more robust alternative to common functions like the squared error, meaning it is less likely to be disproportionally influenced by outliers in the data. The goal of training a network is to find the global minimum of the model's loss equation. A popular method is to adjust the weights and biases descending along the loss function's gradient [27]). In this project, the Nadam [28] optimizer included in Keras [10] is used. It can be classified as a stochastic gradient descent optimizer. By including Nesterov momentum, it is an improvement on the popular Adam [29] (adaptive moment estimation) optimizer. With momentum, the size of the step taken after each iteration of calculating the gradient is increased on the way to a minimum and reduced when getting close to it. This generally accelerates learning, improving overall training time and preventing overshooting a global minimum. The model was trained in epochs of iteration over 2500 training examples. Despite the complexity of the network training, a model for 60 epochs takes only five minutes without any specialized hardware or GPU processing due to the limited amount of training data.

4. Results

For evaluation, 10 models are trained for 75 epochs each. Figure 3 shows that with increasing epochs, the

[10]https://keras.io/optimizers/

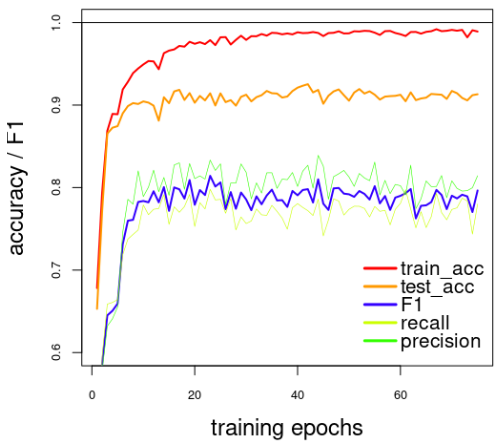

Figure 3. F1, precision, recall, train and test accuracy with increasing training epochs.

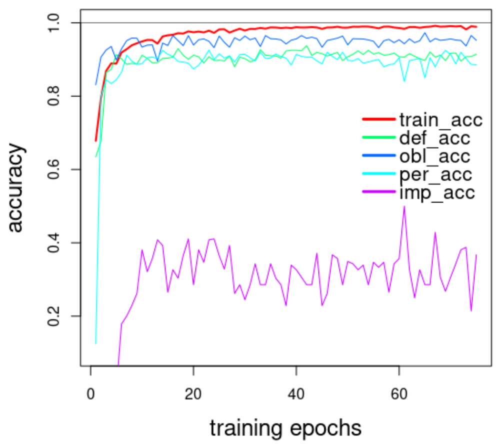

Figure 4. Accuracy of individual classes with increasing training epochs.

training accuracy approaches 100%, increasing only marginally after the 25th epoch. Regarding the test data, however, the accuracy already levels off at around 93% after 20 epochs. With increasing training epochs, the training accuracy increases continually, which does not translate to the testing accuracy due to overfitting. However, since precautions have been taken to minimize the possibility of overfitting, instead of peaking and decreasing, the testing accuracy levels off after reaching its maximum.

Accuracy, giving only an overall assessment of the model's performance, cannot measure how well the model is adapted to the individual classes. In order to see how much the bias in the data affects the model's ability, the harmonic mean between precision and recall or F1 score is calculated [30]. If the model's errors were distributed equally over all classes, the F1 score would equal the overall accuracy. The more unevenly the mistakes are distributed, the lower the F1 score falls below the accuracy. Figure 3 also shows that following the same pattern as training and test, accuracy F1 levels off below 0.8. However, it slightly decreases after epoch 50, which might result from overfitting, favoring classes for which more examples are provided to achieve higher accuracy. In Figure 4 the reason for accuracy and F1 differing can be clearly seen. Likely as the result of too few training examples, the accuracy of the prohibitions class never exceeds 50%. With a total of 104 sentences, prohibitions account for only four percent of the training data, which is seemingly just too little to classify them accurately.

Interestingly, however, permissions, also presenting only a minority of the examples, are classified equally well as the more abundant classes. On the one hand, this could be explained by the fact that permissions are with 250 instances more than twice as frequent as prohibitions. On the other hand, permissions differ linguistically from the other classes more than prohibitions. Whereas prohibitions are often the negation of an obligation or permission, differing only in including a 'not,' permissions are signified by unique clauses like 'shall be authorized to.' Due to this more clear distinction, permissions might require fewer training examples to be correctly classifiable.

In summary, the best model trained using this method achieved a test accuracy of 93.6% and an F1 score of 87.4%. The low F1 score is the result of lacking accuracy in classifying prohibitions. Here it only achieves an accuracy of 58.1%, low compared to the other classes, which all range above 90%.

5. Discussion

5.1. Comparison to Baseline Solution

To facilitate comparability of the new proposed method a simple baseline was established using a Bag of Words[11] (BoW) approach to convert the natural text to a numeric format and a Random Forest Classifier[12] (RFC) to handle the multi-label classification. Instead of vectorizing the pre-processed sentences, the data frame is transformed into a 2800 row long, 5300 column wide sentence-term matrix. By this process, all grammar and sentence structure forms are disregarded in favor of a simple numeric word frequency measure. Finally, a Random Forest model is used in a 5-fold CV to collect the baseline results.

Using the same accuracy and macro-averaged F1 score as with the machine learning model, the baseline

[11]https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.CountVectorizer.html

[12]https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html

method achieves an accuracy of 73% and an F1 score of 0.53. This relatively high, and considering the low F1 score somewhat misleading, accuracy value results from the model devolving into a one rule classifier for two of the four classes. Due to the uneven distribution of positives to negatives for the last two classes, the model arrives at only guessing negatives. This results in a decent accuracy but an abysmal F1 score and highlights the shortcomings of a simple approach like this when confronted with limited training data. However, even for the more readily available classes like 'obligation', the BoW-RFC method presents inferior results, with an accuracy of 72% and an F1 score of 0.63. It sits far behind the accuracy of 96% and F1 of 0.85 for the same class and the CNN Model in our approach. Overall, the proposed new method outperforms the baseline by 20 percentage points in accuracy and an increase in F1 for roughly 0.27.

5.2. Limitations

Most of the limitations result from the limited amount of data and the effort its aggregation entails. Legislative texts do not necessarily follow the patterns of regular speech and writing. Therefore, already available tools for natural language processing (e.g., sentence splitters) are insufficient to preprocess the data independently, and manual processing is necessary. Furthermore, this project serves as a de facto proof of concept, not all sentences of the code of law being included in the data can be justified. In a use-case scenario, this would likely not be an option.

Additionally, it is worth noting that the uneven distribution of classes in the training and test set, which hindered both accuracy and F1 scores, is the result of an uneven distribution thereof in the underlying code of law. Even if more data is collected and more laws are included, the performance for these small classes might not grow in proportion to the performance of other classes. This could lead to a model continually adopting some form of bias when trained on the complete data.

Next, when assigning sentences one of the four discussed labels, a problem presents itself. In German especially, but likely also in other languages, not all obligations stemming from a legal document are presented as obligations in the text. The sentence *"Unternehmen sind steuerpflichtig." (Companies are taxable.)* for example, is written as a definition. Only the adjective "taxable" brings the fact that this entails a duty to pay tax, which makes it an obligation. For this project, all sentences were classified as the deontic modality, so the example sentence above would be labeled as a definition. Finally, it is worth mentioning that we did not perform cross-validation, which is something that needs to be rectified in the future and could possibly result in slightly different results.

5.3. Implications

When discussing the implications of our approach, we can observe that, first, the gathered results show that combining a context-independent vectorization model and a context-dependent classification model is a viable method for sentence classification when confronted with a limited amount of training data. If the data is first transformed into context retaining word vectors, a convolutional neural network can reach competitive levels of accuracy even when trained on very limited data, such as a single code of law. Second, the resulting accuracy of 93.6% is sufficiently high for the output to be used in subsequent rule-based processing steps that can not yet be replaced by machine learning. If similar accuracy can be achieved when applying the same method to other equivalently sized data sets with different classification levels, it could show that this way of pre-labeling sentences can be applied in a wide range of natural language processing tasks.

6. Conclusion and Future Research Steps

This paper presented the first approach for the automated identification and classification of decision rules from legislative texts in the German language using machine learning methods. We created a scalable model using a multi-layer convolutional neural network is created and trained to classify sentences in a code of law based on their deontic expression. A quantitative evaluation showed that our approach achieves high accuracy and precision, with the best model trained using this method reaching a test accuracy of 93.6%. Therefore, we can say that our approach provides a good starting point on top of which rule-based extraction algorithms can easily be built. This would significantly reduce the effort needed for automatic extraction of decision rules from legislative tests, a task that is otherwise manual, complex, and tedious.

In future work, we aim to extend our approach in several directions. First, we aim to conduct a more thorough evaluation of our model compared to other established approaches on the same data set, such are BERT [18] and ELMo [31]. Second, we plan to aggregate more data in hope of reducing the difference in accuracy between classes, as well as to collect data from more than one code of law in order to study how well the algorithm performs when trained on the first, but tested on the second document. Finally, we plan to create a rule-based extraction algorithm on top of this approach, which will be able to automatically visualize extracted decision logic as decision trees or DMN decision tables.

References

[1] A. Rozinat and W. M. van der Aalst, "Decision mining in prom," in *International Conference on Business Process Management*, pp. 420–425, Springer, 2006.

[2] H. van der Aa, C. Di Ciccio, H. Leopold, and H. A. Reijers, "Extracting declarative process models from natural language," in *International Conference on Advanced Information Systems Engineering*, pp. 365–382, Springer, 2019.

[3] L. Arco, G. Nápoles, F. Vanhoenshoven, A. L. Lara, G. Casas, and K. Vanhoof, "Natural language techniques supporting decision modelers," *Data Min. Knowl. Discov.*, vol. 35, no. 1, pp. 290–320, 2021.

[4] V. Etikala, Z. V. Veldhoven, and J. Vanthienen, "Text2dec: Extracting decision dependencies from natural language text for automated DMN decision modelling," in *Business Process Management Workshops*, vol. 397 of *Lecture Notes in Business Information Processing*, pp. 367–379, Springer, 2020.

[5] M. Dragoni, S. Villata, W. Rizzi, and G. Governatori, "Combining NLP Approaches for Rule Extraction from Legal Documents," in *1st Workshop on MIning and REasoning with Legal texts (MIREL 2016)*, (Sophia Antipolis, France), Dec. 2016.

[6] A. Z. Wyner and W. Peters, "On rule extraction from regulations.," in *JURIX*, vol. 11, pp. 113–122, 2011.

[7] J. Zhang and N. M. El-Gohary, "Semantic nlp-based information extraction from construction regulatory documents for automated compliance checking," *Journal of Computing in Civil Engineering*, vol. 30, no. 2, p. 04015014, 2013.

[8] T. M. van Engers, R. van Gog, and K. Sayah, "A case study on automated norm extraction," *Legal Knowledge and Information Systems. Jurix*, pp. 49–58, 2004.

[9] Y. Kim, "Convolutional neural networks for sentence classification," *CoRR*, vol. abs/1408.5882, 2014.

[10] M. M. Lopez and J. Kalita, "Deep learning applied to NLP," *CoRR*, vol. abs/1703.03091, 2017.

[11] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," 2013.

[12] G. Governatori, "Representing business contracts in ruleml," *International Journal of Cooperative Information Systems*, vol. 14, no. 02n03, pp. 181–216, 2005.

[13] J. P. C. J. J. D. W. Loos, Eugene E.; Susan Anderson; Dwight H. Day, *Glossary of linguistic terms*, ch. What is a deontic modality? SIL International, 2009.

[14] C. Prakash, P. K. Chittimalli, and R. Naik, "Open information extraction using dependency parser for business rule mining in sbvr format," in *14th Innovations in Software Engineering Conference (formerly known as India Software Engineering Conference)*, pp. 1–11, 2021.

[15] P. Quaresma and T. Gonçalves, "Using linguistic information and machine learning techniques to identify entities from juridical documents," in *Semantic Processing of Legal Texts*, pp. 44–59, Springer, 2010.

[16] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov, "Enriching word vectors with subword information," 2016.

[17] E. Elwany, D. Moore, and G. Oberoi, "BERT goes to law school: Quantifying the competitive advantage of access to large legal corpora in contract understanding," *CoRR*, vol. abs/1911.00473, 2019.

[18] J. Devlin, M. Chang, K. Lee, and K. Toutanova, "BERT: pre-training of deep bidirectional transformers for language understanding," *CoRR*, vol. abs/1810.04805, 2018.

[19] P. McNamara, "Deontic logic," in *Handbook of the History of Logic*, vol. 7, pp. 197–288, Elsevier, 2006.

[20] Y. LeCun, Y. Bengio, *et al.*, "Convolutional networks for images, speech, and time series," *The handbook of brain theory and neural networks*, vol. 3361, no. 10, p. 1995, 1995.

[21] D. Ciregan, U. Meier, and J. Schmidhuber, "Multi-column deep neural networks for image classification," in *2012 IEEE conference on computer vision and pattern recognition*, pp. 3642–3649, IEEE, 2012.

[22] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems 25* (F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, eds.), pp. 1097–1105, Curran Associates, Inc., 2012.

[23] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov, "Improving neural networks by preventing co-adaptation of feature detectors," 2012.

[24] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *Proceedings of the 27th international conference on machine learning (ICML-10)*, pp. 807–814, 2010.

[25] A. Jacovi, O. S. Shalom, and Y. Goldberg, "Understanding convolutional neural networks for text classification," 2018.

[26] P. J. Huber, "A robust version of the probability ratio test," *The Annals of Mathematical Statistics*, vol. 36, no. 6, pp. 1753–1758, 1965.

[27] S. Ruder, "An overview of gradient descent optimization algorithms," *CoRR*, vol. abs/1609.04747, 2016.

[28] T. Dozat, "Incorporating nesterov momentum into adam," *ICLR*, 2016.

[29] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *ICLR (Poster)*, 2015.

[30] C. J. van Rijsbergen, "A new theoretical framework for information retrieval," *SIGIR Forum*, vol. 51, no. 2, pp. 44–50, 2017.

[31] M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer, "Deep contextualized word representations," in *NAACL-HLT*, pp. 2227–2237, Association for Computational Linguistics, 2018.