NOMOS: Navigating Obligation Mining in Official Statutes

Andrea Pennisi[*,**], Elvira González Hernández[*], and Nina Koivula[*]

[*]Enhesa S.A., 1050 Brussels, Belgium
[**]Department of Mathematics, Computer Science, and Economics,
University of Basilicata, 85100 Potenza, Italy
name.surname@enhesa.com

Abstract

The process of identifying obligations in a legal text is not a straightforward task, because not only are the documents long, but the sentences therein are long as well. As a result of long elements in the text, law is more difficult to interpret (Coupette et al., 2021). Moreover, the identification of obligations relies not only on the clarity and precision of the language used but also on the unique perspectives, experiences, and knowledge of the reader. In particular, this paper addresses the problem of identifying obligations using machine and deep learning approaches showing a full comparison between both methodologies and proposing a new approach called NOMOS based on the combination of Positional Embeddings (PE) and Temporal Convolutional Networks (TCNs). Quantitative and qualitative experiments, conducted on legal regulations [1], demonstrate the effectiveness of the proposed approach.

1 Introduction

In the legal context, obligations can be defined as "a legal constraint imposed by law and addressed to a juridical person" (Iacono et al., 2021). They are often enforceable and backed by legal consequences in case of non-compliance establishing the rights and duties between parties, and regulating their interactions within a legal framework. In other words, an obligation represents a commitment to perform a specific action or refrain from certain behaviors. The interpretation of an obligation is not straightforward because it depends on several elements : the clarity of the text, the precision of the language, and moreover, the evaluation, the knowledge, and the experience of the reader. Consequently, two or more individuals may perceive different obligations when presented with the same legal text. Besides, even the most straightforward legal obligations are not always self-standing, and often encompass several sentences. They can also be fragmented on the paragraph level, whereby the paragraph begins with a generic phrase, followed by several sub-paragraphs, each of which introduces a different obligation that still relies on the initial generic phrase for correct interpretation. This sub-paragraph structure creates complications for traceability (Kiyavitskaya et al., 2008).

In this paper, we present a comparison of several machine learning and deep learning methodologies for extracting obligations together with a solution based on the combination of Positional Embeddings (PE) and Temporal Convolutional Networks (TCNs). The contribution of this work is two-fold.

1. A full comparison between machine learning and deep learning methodologies on a set of different legislation containing a variety of obligations.
2. NOMOS, a network based on the use of a PE layer able to encode the position of the embeddings in order to maintain a temporal relation between words, and a TCN for extracting the main features from the embeddings and classifying the sentences.

The remainder of the paper is organized as follows. Section 2 contains a description of the existing methods for extracting obligations in the legal domain. Section 3 presents the methodologies for identifying obligations and the solution we adopted. Quantitative and qualitative experimental results are shown in Section 4. Finally, conclusions are drawn in Section 5.

2 Related Work

In the last years, the automatic analysis of legal texts has gained significant attention, and numerous studies and approaches have emerged that are capable of dealing with classifying complex legal texts. Glaser et al. (2018) carried out experiments to classify legal clauses and sentences in

1. Regulation (EU) 2023/988 ('General Product Safety Regulation' or 'GPSR')

Proceedings of the Natural Legal Language Processing Workshop 2023, pages 8–16

the German Civil Code, legal texts in a different domain, and rental contracts, to investigate the portability of their models. They selected a taxonomy of 9 classes for classifying legal sentences, including for example Duty, Prohibition, Definition, and Consequence, the system being based on the function of the sentence for the subject of the law. The dataset consisted of 913 sentences that had been manually classified by legal professionals. They chose 6 types of models : Multinomial Naive Bayes (MNB), Linear Regression (LR), Support Vector Machine (SVM), Multilayer Perceptron (P), Random Forests (RF), and an Extra Trees Classifier (ETC). The limitations of this study include the contract law focus – contract datasets being generally more restricted in scope and variation when compared to regulatory datasets – and the many semantic classes of legal sentences applied on a relatively small dataset.

In de Maat et al. (2010), the authors evaluated traditional machine learning methods, namely SVMs, against a knowledge-based method of standard phrases. Their dataset was divided into 13 classes of sentences, including definitions, permissions, enactment dates, and obligations. While they were able to reach over 90% accuracy for recall and precision with the SVM, the authors found that it did generalize well compared to the pattern-based classifier on laws that were not included in the training set. This led them to conclude that the pattern-based classifier is the better choice because the classification criteria are transparent and customizable. It should be noted that only 181 out of 584 sentences were obligations.

On the other hand, Iacono et al. (2021), trained a neural network classifier for legal obligations found in Italian law. Their classifier contained fewer categories (i.e. : no obligation, relevant obligation, and irrelevant obligation, based on their impact on financial institutions) and the dataset contained 10.628 clauses (the distribution of the categories within the training clauses is not provided). The labels for the training corpus were provided by non-expert annotators. Their model of choice was the pre-trained UmBERTo2 [2], to which they added a ReLu activation function for classification. The results showed an unreliable performance in distinguishing between relevant and non-relevant obligations, but significantly improved accuracy on the binary task of obligation versus no obligation, reaching an F-score of 0.97 for detecting not obligations and 0.91 for detecting obligations. However, the performance of the non-expert annotators in distinguishing between relevant and irrelevant obligations was also evaluated and found lacking. An additional limitation of this study was the absence of a validation dataset employed by the authors to evaluate the performance of their model.

In O'Neill et al. (2017), the authors performed a study on classifying sentences in financial regulations. In this study, the extraction of deontic modalities from regulatory texts was treated as a probabilistic rather than a logical problem. The training dataset was labeled by expert annotators and divided into 596 obligations, 607 permissions, and 94 prohibitions. For testing, the authors relied on a gold standard test set with separate documents of EU and UK financial laws. Then, they used word2vec embeddings specific to the legal domain coupled with pre-trained Google News embeddings as input to the classifiers. The authors compared a set of traditional machine learning algorithms, including LR and SVM, to a set of deep learning algorithms such as LSTM, CNN-LSTM, and BiLSTM. When applying feature transformation and Information Gain (IG) feature selection on Google News embeddings, SVM outperformed all the other traditional methods on the test set but only reached an accuracy of 56.12%. On the other hand, the ANN models produced significantly better results than SVM, with two versions of BiLSTM reaching an accuracy of over 81%. However, the results are not aggregated per sentence classes across the board, making it difficult to know what the performance is when it comes to detecting obligations in particular.

The use of Temporal Convolutional Networks (TCNs) in the legal domain is not yet widespread. In the state-of-the-art, there are just a few examples of the use of TCNs for generic sentence classification tasks. For example, in Zuo et al. (2020), the authors described a bidirectional temporal convolutional network combined with an attention mechanism for sentence classification, reaching an accuracy rate of 91.47% on a public dataset. The main issue of such an approach is related to its effectiveness only in classifying short sentences like news.

In this work, we propose a comparison of 3 state-of-the-art approaches, namely : Tf-Idf and SVM, TCN, and BERT. In our analysis, we focus only on TCN over LSTM because of its ability to automatically extract features without the use of any

2. https://huggingface.co/Musixmatch/umberto-wikipedia-uncased-v1

Convolutional layer, and moreover, because TCN possesses very long effective history sizes allowing the network to look very far into the past to make a prediction. We also present a novel approach based on the combination of Positional Embeddings and TCN that we adopted for detecting obligations in law texts.

3 Methodologies

In Sec. 2, we described the common approaches that can deal with obligation extraction. In our study, we selected and compared 3 methodologies, namely : 1) Term Frequency-Inverse Document Frequency (Tf-Idf) and Support Vector Machine (SVM), 2) Temporal Convolutional Neural Networks, and 3) Bidirectional Encoder Representations from Transformers (BERT). Moreover, we propose our solution for obligation classification. Each methodology is detailed in the sections below.

3.1 Tf-Idf and SVM

Term Frequency-Inverse Document Frequency (Tf-Idf) is a method that measures how relevant a term is within a document relative to a collection of documents (i.e. corpus). It proportionally increases the frequency of the appearance of a word in a document, and balances it by the number of documents that include the word. So, words that are very common in every document have a low rank even if they appear many times but it is likely that they are meaningless for the analysis of the document.

Tf-Idf is calculated by multiplying two metrics : 1) *Term frequency* (*Tf*) : which calculates how many times a word appears in a document, and 2) *Inverse document frequency* (*Idf*) : which computes the inverse document frequency of a word across a set of documents.

Given a term t and a document d, *Tf* is defined as follows :

$$tf(t,d) = \frac{f_{t,d}}{\sum_{t' \in d} f_{t',d}} \quad (1)$$

Where $f_{t,d}$ is the number of times that term t occurs in the document d. The denominator represents the occurrence of the same terms t' in the document d.

While Idf is the logarithmically scaled inverse fraction of the documents that contain the word :

$$idf(t,D) = \frac{N}{1 + |\{d \in D : t \in d\}|} \quad (2)$$

Where :

— D is the corpus ;
— N is the total number of documents in the corpus $N = |D|$;
— $|\{d \in D : t \in d\}|$ represents the number of documents where the term t appears. We add 1 to the denominator because if $t \notin D$, the previous expression leads to a zero division.

Then, given the Eq. 1 and 2, Tf-Idf is calculated as :

$$tf\text{-}idf(t,d,D) = tf(t,d) \times idf(t,D) \quad (3)$$

A high-term frequency and a low-term document frequency in the whole collection lead to a high weight in Tf-Idf, hence filtering out the common terms. In Natural Language Processing (NLP), Tf-Idf is usually used with a classical machine learning technique to deal with sentence classification problems. For this work, our decision fell on Support Vector Machine (SVM). As shown in Glaser et al. (2018), combining Tf-Idf and SVM is successful in classifying semantic types of legal sentences.

SVM can handle high-dimensional data like the one corresponding to text and perform well with small datasets, as it requires a small number of support vectors to define the boundary. SVM can model non-linear decision boundaries by using the kernel trick, which maps the data into a higher-dimensional space where the data becomes linearly separable. Moreover, it uses only a subset of the training data to make predictions, which makes it very efficient and less prone to overfitting.

3.2 TCN

Temporal Convolutional Network (TCN) proposed by Bai et al. (2018) is a dilated-causal version of Convolutional Neural Network (CNN). Often Recurrent Neural Networks (RNNs) (Zaremba et al., 2014), Long Short-Term Memory networks (LSTMs) (Hochreiter and Schmidhuber, 1997), and Gated Recurrent Unit networks (GRUs) (Chung et al., 2014) are associated with sequence modeling tasks. This process is usually performed in 2 steps : 1) low-level feature extraction using a CNN that encodes the spatial-temporal information, and 2) one or more RNN (LSTM, or GRU) layers that capture high-level temporal information. The main disadvantage of such an approach is that it requires two separate models. TCN captures both levels of information hierarchically.

As mentioned in Bai et al. (2018), the main features of TCNs are : 1) the dilated-causal convo-

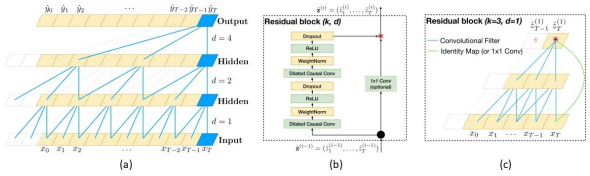

FIGURE 1 – TCN Component Architecture (Bai et al., 2018). (a) A dilated casual convolutional neural network. (b) TCN residual block. (c) An example of residual connections.

lution layers in the architecture do not share any information from future to past (Fig. 1a) and 2) the architecture manages sequences of any lengths and maps them to an output sequence of the same length (Fig. 1b and c). Fig. 2 shows the architecture of the network used for the methodology comparison. The dilated-causal convolutional layer is able to let the network look back up to $(k-1)d$ time steps (where d is the number of steps) to achieve an exponentially larger receptive field with fewer parameters and layers. d is exponentially increased with the depth of the network : $d = O(2^i)$, where $i = 0, \cdots, n$ is the level of the network. Fig. 1a shows how the dilated-causal convolution on the first hidden layer is applied every two steps where $i = 1$. The arrangement of the dilated-causal convolutional layers ensures that some filter is applied to each input within the history allowing a long effective history. Fig. 1b shows the residual block (He et al., 2016) of TCN composed of 2 dilated-causal convolutional layers, weight normalization, ReLU activation, and dropout. It is worth noting that, for the correct execution of the residual connection, there is an optional 1×1 convolution layer that is applied if the number of input channels differs from the number of the output channels from the dilated causal convolutional layers.

3.3 BERT

Bidirectional Encoder Representations from Transformers (BERT) is a pre-trained deep bidirectional representation from the unlabeled text by joint conditioning on both the left and right context. The architecture consists of several transformer encoders stacked together, which encapsulate two sub-layers : 1) a self-attention layer and 2) a feed-forward layer. Fig. 3 shows the transformer architecture, BERT is based only on the encoder part. In order to process the text sequence all at once, the transformer adds, in addition to the standard word embedding layer, a special embedding layer with positional information to the standard encoding system together with a segment layer to distinguish between two or more consecutive sentences (Fig. 4). The maximum size of tokens that can be fed into the BERT model is 512. The attention mechanism focuses on the important parts of the information blending out the unimportant ones and it can be considered as a mapping between a query and a set of key-value pairs to an output.

Vaswani et al. (2017) propose a scaled dot-product attention with multi-head attention on top of it, which uses 3 matrices, namely : query (Q), keys (K), and values (V), representing different projections of the same input sentence. Therefore,

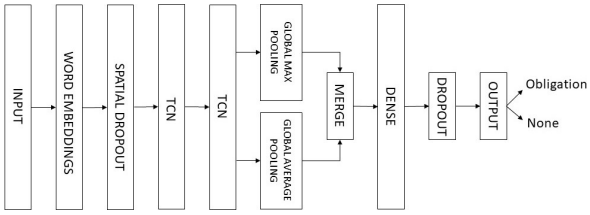

FIGURE 2 – TCN Network for sentence classification (Henkel, 2021).

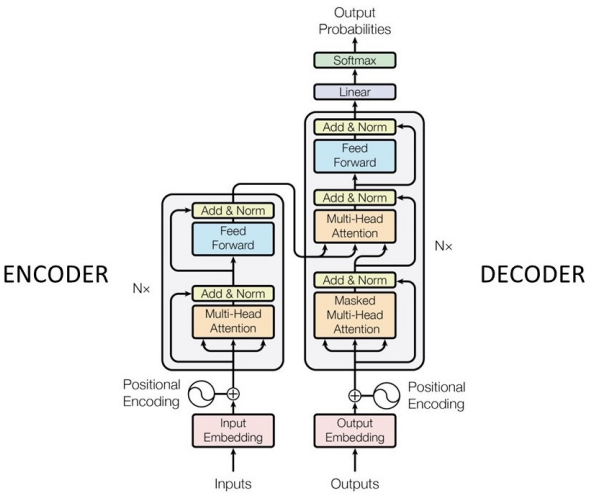

FIGURE 3 – Transformer Architecture (Vaswani et al., 2017).

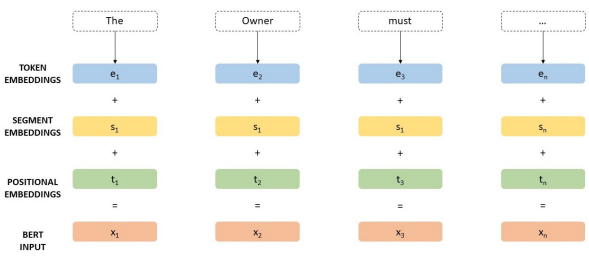

FIGURE 4 – Transformer Input Embeddings.

it implements a self-attention mechanism by capturing the relationships between the different words of the same sentence. The self-attention mechanism can be formulated as follows :

$$attention(Q, K, V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{4}$$

where : d_k is the dimension of the vectors q and k containing the queries and keys, respectively. The multi-head attention mechanism uses h linear learned projections of Q, K, and V. Then, the single attention mechanism is applied to each projection in parallel to produce outputs that, in turn, are concatenated and projected to produce the final result. In such a way, it allows the attention function to extract information from different representations instead of using a single one. The multi-head attention function can be defined as follows :

$$multi\text{-}head(Q, K, V) = concat(head_1, \cdots, head_n)W^O \tag{5}$$

where $head_i = 1, \cdots, h$ implements a single attention function characterized by its own learned projection matrices, and W^O is the weight matrix.

In order to have a deeper sense of language context, BERT uses bidirectional training that takes both the previous and next tokens into account simultaneously. BERT applies the bidirectional training of the transformer to language modeling and learns the text representations. Intuitively, the encoder returns the same input vectors but *augmented* with more complex information. Since, BERT aims at predicting words using the so-called Masked LM (MLM) by randomly masking words in the sentence and then predicting them, in order to accomplish the task of sentence classification, we add one dense layer on top of BERT's output to perform the sentence classification.

3.4 Proposed Method

Fig. 5 shows the architecture of NOMOS. The network uses the sum of Positional and Word embeddings to encode the sentence into the network. The positional layer brings the advantage of encoding the position of the words in each sentence, which allows the network to be able to understand the pattern from the encoding and generalize better for longer sentences like those in legal texts. The rest of the network is inspired by the network presented in Henkel (2021), with the exception that we used a single TCN layer. Sec. 4.2 describes the parameters we used for training the network and how we found them.

4 Experimental Results

To evaluate all the approaches we described in Sec. 3, a dataset of 74825 sentences has been labeled by expert lawyers and it is composed of 35850 obligations and 38975 not obligations. The sentences have been extracted from the Code of Federal Regulations [3] and Australasian Legal Information Institute [4]. We divided the dataset according to the rule : 70% as the training set, 20% as the testing set, and 10% as the validation set. We used the same sentence splitter we applied for preprocessing the text in all the methodologies we proposed for this work to split the texts into sentences.

For training and testing, we used a laptop running Ubuntu 22.04 LTS and equipped with an Intel i9-12950HX CPU with 32 GB RAM and an NVIDIA RTX A5500. We used Tensorflow [5] for developing the Deep Learning algorithms, while we relied on Scikit-learn [6] for the machine learning ones.

4.1 Text Preprocessing

We noticed that often a legal text includes lists and periods not related to the end of the sentence that the standard methods based on libraries such as Spacy [7] or NLTK [8] are not able to manage. Therefore, we developed a custom-made sentence splitter methodology that we used for all the described approaches.

Our preprocessing method is based on detecting all the periods in the text and, by using a set of rules, understanding if the period is real or just a dot that separates numbers, numbered lists, etc. In such a way, we are able to deal with any list and period in the text and correctly split it into sentences. Then, for each sentence, we remove all the stop words and perform Tf-Idf for the SVM model and the tokenization method provided by Tensorflow [9] for the deep learning models.

3. https://www.govinfo.gov/help/cfr
4. https://www.austlii.edu.au/
5. https://www.tensorflow.org/
6. https://scikit-learn.org/stable/
7. https://spacy.io/
8. https://www.nltk.org/
9. https://shorturl.at/nFVXY

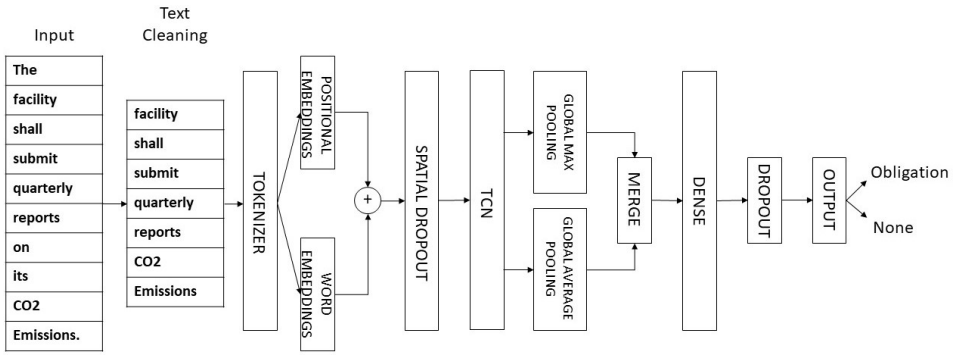

FIGURE 5 – The architecture of NOMOS.

Method	Parameters				
SVM	**N-GRAMS**	**C**	**GAMMA**	**KERNEL**	
	(3, 3)	100	0.01	RBF	
TCN	**SPATIAL DROPOUT**	**TCN FILTER #1**	**TCN FILTER #2**	**DENSE FILTER**	**DROPOUT**
	0.1	128	64	32	0.1
NOMOS	**SPATIAL DROPOUT**	**#TCN LAYERS**	**TCN FILTER**	**DENSE FILTER**	**DROPOUT**
	0.2	1	128	32	0.1

TABLE 1 – The parameters used for training all the algorithms.

4.2 Training Parameters

To find the right parameters of the SVM, we used a grid-search cross-fold validation approach. We included in the grid the possibility of using n-grams for Tf-Idf, multiple kinds of kernels, and different values of C (the regularization parameter) and $gamma$ (the kernel coefficient). While for the TCN network described in Sec. 3.2, we used a Bayesian Optimization (Snoek et al., 2012) technique combined with cross-fold validation for finding the following parameters : Spatial Dropout probability, TCN filters, Dense filters, and Dropout probability. For our approach, we used the same Bayesian Optimization technique but we included the number of TCN layers as a parameter. The F1-score (see Sec. 4.3) metric, representing the harmonic mean between precision and recall, has been used as a function to maximize in the optimization process. As a network optimizer, we used AdaMax (Kingma and Ba, 2017) since it is more suitable for sparsely updated parameters (e.g. embeddings). BERT has been used with the standard parameters, we added only a classification layer at the end. In particular, we used the largest model of BERT [10]. Tab. 1 shows a summary of the best parameters found during the optimization process.

10. https://huggingface.co/google/bert_uncased_L-12_H-768_A-12

4.3 Performance Metrics

To evaluate the goodness of the methodologies, we compared the predictions and the ground truth sets. In our quantitative comparison, we computed true positive (TP), false positive (FP), true negative (TN), and false negative (FN) sets in order to calculate the following metrics :

— $precision = \frac{TP}{TP+FP}$,
— $recall = \frac{TP}{TP+FN}$,
— $F1\text{-}score = \frac{2 \times precision \times recall}{precision+recall}$.

In the next section, we discuss the results we obtained from a qualitative and a quantitative point of view.

Method	Precision	Recall	F1-Score
Tf-Idf+SVM	0.94	0.93	0.93
TCN	0.94	0.94	0.94
BERT	**0.95**	**0.95**	**0.95**
NOMOS	**0.95**	**0.95**	**0.95**

TABLE 2 – Quantitative Results.

4.4 Quantitative Results

Tab. 2 shows the results we obtained in our experiments. BERT and NOMOS reach the highest

values for all the metrics we considered. For such a reason, we made a further analysis only considering these two algorithms to understand how good they are in distinguishing between the two classes.

Tab. 3 and Tab. 4 show that NOMOS performs better in identifying the obligations, this is highlighted even in Fig. 6, representing the confusion matrices of both the methods.

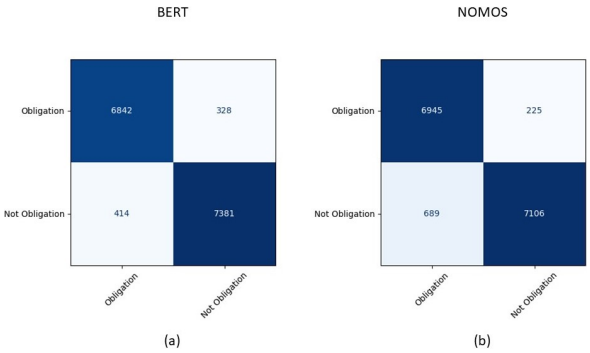

FIGURE 6 – Confusion Matrices of (a) BERT and (b) NOMOS.

Our model is able to provide very effective results using only 2.5 million parameters compared to the 28.7 million parameters of BERT.

4.5 Qualitative Results

All the models have been tested on a reference text, namely : Regulation (EU) 2023/988 [11] and evaluated by an expert. The text includes 643 sentences, 88 of which are obligations. Such an evaluation allows us to understand how well the models perform and generalize on texts not belonging to the dataset used for training and testing them.

Tf-Idf and SVM produces a significant number of false positives. On the test regulation, it finds 205 obligations when there are only 88. Some of them follow a clear pattern, while other false positive sentences do not contain any clues as to why they have been labeled as obligations. For example, the model is not able to infer the context around obligation words. Preambular text such as «*Directive 2001/95/EC of the European Parliament and of the Council (3) lays down the requirement that consumer products must be safe and that Member States' market surveillance authorities must take action*» does contain deontic language but it is not in itself an obligation. Several similar sentences are falsely labeled as an obligation. The other pitfall of this model is false positive statements containing the verb "may", where the statements express permission rather than an obligation. For example, the following sentence is a false positive : «*For that purpose, Member States may notify through the Safety Gate Rapid Alert System corrective measures taken by their authorities or by economic operators on the basis of this Regulation, of Union harmonization legislation and of Regulation (EU) 2019/1020 in relation to products presenting a less than serious risk.*» Lastly, the performance of the model is inconsistent. For example, «*The Commission shall draw up guidelines for the practical implementation of the Safety Business Gateway*» was labeled as an obligation while «*The Commission shall inform consumers and other interested parties of its action*» was not.

TCN finds 153 obligations out of 88 true obligations, which is already a clear improvement from the user perspective compared to the SVM. However, it also suffers from excessive false positives as well as false negatives. The problematic patterns can be traced back to the verbs “ensure”, “may” and “provide”, which all feature prominently in the false positives. For example, the sentence «*Product identification and the provision of information on the manufacturer and other relevant economic operators thus ensures that consumers, including persons with disabilities, and market surveillance authorities obtain accurate information regarding dangerous products, which enhances confidence in the market and avoids unnecessary disruption of trade*» is falsely labeled as an obligation. There are also several false negatives containing the verb “shall”, which is highly problematic because this is a common indicator of a true obligation. An example of this error is the following sentence, which was labeled as a non-obligation : «*The economic operator referred to in paragraph 1 of this Article shall, upon request by the market surveillance authori-*

11. https://eur-lex.europa.eu/legal-content/EN/TXT/PDF/?uri=CELEX:32023R0988

Class	Precision	Recall	F1-Score
Obligation	0.93	0.96	0.945
Not Obligation	0.96	0.93	0.945

TABLE 3 – BERT results for each class.

Class	Precision	Recall	F1-Score
Obligation	0.94	0.96	0.95
Not Obligation	0.96	0.94	0.95

TABLE 4 – The results for each class using NOMOS.

ties, provide documented evidence of the checks performed». Lastly, like the SVM, this model also suffers from inconsistency where similar sentences receive an opposite label, meaning that the trust of the user is negatively impacted.

BERT labels 117 sentences as obligations, a number which is not significantly higher than the expected amount (88). However, it proves to be unreliable in the real world, because a large amount of these are incorrect. In particular, the model mislabels a large number of sentences that contain the modal verb "should", which expresses a wish or a principle rather than an obligation and is overwhelmingly found in the non-binding preambular text of regulatory documents. An example of this includes the following false positive : «*Under the general safety requirement laid down in this Regulation, economic operators should be obliged to place only safe products on the market.*» This means that the model is not dependable for professional use. At the same time, it is not consistent, as the following sentence is not labeled as an obligation : «*The technical documentation should be based on an internal risk analysis carried out by the manufacturer.*»

NOMOS finds 159 obligations in the test document with 88 true obligations. While the model is still over-inclusive to some extent, it produces fewer false negatives compared to the other types of models, which is an important characteristic for the user to trust the model. At the same time, it also does not produce false positives (e.g. : «*That requirement shall not apply where the product can be used safely and as intended by the manufacturer without such instructions and safety information.*») due to misunderstanding the context around the obligation words, meaning that it is the most robust algorithm from the user perspective.

4.6 Discussion

Both qualitative and quantitative results show the effectiveness of NOMOS in detecting obligations. In particular, in the qualitative results, we showed how the model deals with a different country's legislation not included in the training dataset. However, the method has 2 limitations : 1) it does not perform well with passive English obligations (i.e. sentences where the subject to which the obligation refers, is not explicitly expressed), and 2) the method is not sufficiently reliable in distinguishing between obligations directed at private parties and the ones directed at authorities.

5 Conclusions

In this work, we have presented a comparison of methodologies for extracting obligations from legal texts. Apart from showing well-known methodologies, we proposed our own solution called NOMOS for dealing with such a classification problem based on the combination of positional and word embeddings and a Temporal Convolutional Network. Quantitative and qualitative results demonstrated the effectiveness of our approach in distinguishing obligations from non-obligation sentences reaching a $F1\text{-}score$ on both classes of 0.95.

As future directions, we intend to introduce an attention layer to the network in order to focus on the subject, the verb, and the object of a sentence and distinguish between obligations directed at private parties and those directed at authorities.

References

Shaojie Bai, J. Zico Kolter, and Vladlen Koltun. 2018. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. *CoRR*, abs/1803.01271.

Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling.

Corinna Coupette, Dirk Hartung, Janis Beckedorf, Maximilian Böther, and Daniel Martin Katz. 2021. Law smells : Defining and detecting problematic patterns in legal drafting. *CoRR*.

Emile de Maat, Kai Krabben, and Radboud Winkels. 2010. Machine learning versus knowledge based classification of legal texts. In *Proceedings of the 2010 Conference on Legal Knowledge and Information Systems : JURIX 2010 : The Twenty-Third Annual Conference*, pages 87–96.

Ingo Glaser, Elena Scepankova, and Florian Matthes. 2018. Classifying semantic types of legal sentences : Portability of machine learning models. In *Legal Knowledge and Information Systems - JURIX 2018 : The Thirty-first Annual Conference, Groningen, The Netherlands, 12-14 December 2018*, Frontiers in Artificial Intelligence and Applications.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep Residual Learning for Image Recognition. In *Proceedings of 2016 IEEE Conference on Computer Vision and Pattern Recognition*.

Christof Henkel. 2021. Temporal Convolutional Network. https://www.kaggle.com/code/christofhenkel/temporal-convolutional-network.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*.

Maria Iacono, Laura Rossi, Paolo D'Angelo, Andrea Tesei, and Lorenzo De Mattei. 2021. An obligations extraction system for heterogeneous legal documents : Building and evaluating data and model. In *Proceedings of the Eighth Italian Conference on Computational Linguistics, CLiC-it 2021, Milan, Italy, January 26-28, 2022*, volume 3033 of *CEUR Workshop Proceedings*. CEUR-WS.org.

Diederik P. Kingma and Jimmy Ba. 2017. Adam : A method for stochastic optimization.

Nadzeya Kiyavitskaya, Nicola Zeni, Travis D. Breaux, Annie I. Antón, James R. Cordy, Luisa Mich, and John Mylopoulos. 2008. Automating the extraction of rights and obligations for regulatory compliance. In *Conceptual Modeling - ER 2008*.

James O'Neill, Paul Buitelaar, Cecile Robin, and Leona O' Brien. 2017. Classifying sentential modality in legal language : A use case in financial regulations, acts and directives. In *Proceedings of the 16th Edition of the International Conference on Artificial Intelligence and Law*, ICAIL '17, page 159–168.

Jasper Snoek, Hugo Larochelle, and Ryan P. Adams. 2012. Practical bayesian optimization of machine learning algorithms.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*. Curran Associates, Inc.

Wojciech Zaremba, Ilya Sutskever, and Oriol Vinyals. 2014. Recurrent neural network regularization.

Ying Zuo, Lifen Jiang, Huazhi Sun, Chunmei Ma, Yan Liang, Shuaibao Nie, and Yongheng Zhou. 2020. Short text classification based on bidirectional tcn and attention mechanism. *Journal of Physics : Conference Series*.