Deep Learning-Based Entity Recognition in Construction Regulatory Documents

Phillip Schönfelder and Markus König

Department of Civil and Environmental Engineering, Ruhr University Bochum, Germany

phillip.schoenfelder@rub.de, koenig@inf.bi.rub.de

Abstract -

In the construction industry, contractors require precise knowledge of design restrictions originating from regulatory documents and contract specifications. For the automatic compliance checking of the building design regarding these rules, they have to be converted from the representation in natural language to a machine-readable format. This task, if carried out by human experts, is quite laborious and error-prone, and thus its automation is anticipated. A building block of this information extraction process is to find the key terms which carry the semantic information in each design rule. Named entity recognition, a sub-task of information extraction in the field of natural language processing, aims towards finding these entities in unstructured text and assigning them a label according to predefined classes. This paper presents a method based on a supervised deep learning transformer model, which is used to extract relevant terms from a corpus of German regulatory documents. It requires few training data, no user interaction and achieves weighted performance scores of over 95% precision and 95% recall, given that 12 unbalanced classes are specified. Additionally, it is investigated how different tagging schemes and model variations affect the classifier's performance. For future extensions, the class labels are chosen such that they can be linked to concepts already defined by Industry Foundation Classes. As part of this study, a training data set is created consisting of 2500 sentences from construction law documents, annotated with named entity tags.

Keywords -

automatic compliance checking, building information modeling, natural language processing, named entity recognition, machine learning

1 Introduction

Building information modeling (BIM) offers great potential for the automation of processes in the architecture, engineering and construction (AEC) industry. One of these processes is the checking of the building design against rules and specifications from law documents, technical guidelines and individual contract requirements. For example, the building design must ensure accessibility and meet appropriate fire safety measures, according to the respective regulations.

Automatic compliance checking (ACC) requires a BIM model to be checked on the one hand, and rule specifications in machine-readable format on the other. Regulatory documents are however written in natural language, and thus the rules must be transformed into logical, structured expressions to be processed automatically. Obtaining such logical expressions from these unstructured text documents is however a quite complex task. If carried out manually by domain experts, it turns out to be error-prone and time-consuming. Hence, the objective is to develop automated systems that perform rule extraction with a minimum of human interaction.

An important building block in this process is the extraction of relevant terms from the clauses found in legal documents pertaining to buildings, building parts, documents, organizations, material types, and other engineering concepts. These terms are called *entities* in the realm of named entity recognition (NER), a sub-task of information extraction (IE) in the field of natural language processing (NLP). Subsequent processes, such as the extraction of relations between entities or finding semantic triplets rely on the priorly detected named entities. [1]

Previous approaches of NER in the construction domain include Li et al. [1], who used a Bi-LSTM architecture with a self-attention mechanism to detect entities as part of their relation extraction procedure. Zhang and El-Gohary [2], renowned researchers in ACC, proposed an unsupervised-learning approach to link terms in the legal text to IFC concepts based on semantic similarity. As a numerical measure, they used the cosine similarity of the word embeddings. Recently, Moon et al. [3] used an architecture consisting of a Bi-LSTM and an on-top conditional random field (CRF) layer for NER in a variety of English regulatory documents, mainly standard specifications of US states. For the proposed six distinct entity classes, their method achieved performance scores of 91.9% precision and 91.4% recall. A similar architecture was employed by Song et al. [4], who report precision and recall scores of 39% and 68%, respectively, for a dataset in Chinese language. Zhong et al. [5] propose an end-to-end neural architecture to extract temporal constraints from Chi-

nese building codes. One of their involved process steps is to identify processes, objects and interval times by a NER algorithm based on a Bi-LSTM + CRF architecture. While many of the stated approaches achieve high performance scores and allow for the integration into promising algorithms, none of them has taken advantage of state-of-the-art transformer model architectures. However, for the purpose of classifying near-miss safety reports of the construction industry, transformer models have already been employed successfully by Fang et al. [6]. In this paper, a method is presented to find entities in unstructured regulatory construction documents and classify them with regard to selected Industry Foundation Classes (IFC) and a handful of generic classes. To achieve this goal, a state-of-the-art deep learning transformer model is trained with an annotated training set consisting of German public construction law texts.

The remaining part of the paper proceeds as follows: Section 2 lays out some of the most important background information about the tools used for this paper. Section 3 is concerned with data processing and presents how the existing transformer model can be extended to suit the task at hand. Section 4 addresses the experimental setup and includes detailed information about the learning algorithm. The results of the conducted experiments are displayed in Section 5. The findings are wrapped up in Section 6.

2 Background

2.1 Transformer models in NLP

Natural language is a form of sequential data and thus its processing requires adequate handling of data series. A frequently used tool in this regard are recurrent neural networks (RNNs), which inherently have the ability to process input data sequentially. With the advent of transformer models [7] in 2017, RNNs are being superseded by this new model architecture in many applications. Transformer models have the distinct advantage of being efficiently parallelizable and thus they can make use of GPUs, which reduces training times. This makes training on huge amounts of data possible.

A prominent example of transformer models is BERT (Bidirectional Encoder Representations from Transformers) [8]. Roughly speaking, it adapts the encoder part of the transformer architecture and stacks a total of twelve encoders, each with a hidden dimension of $H = 768$ per input token. This makes up to a huge model with roughly 1.1×10^8 weights. A high-level view of the architecture is displayed in Figure 1. At first, the input sentence is tokenized with respect to a fixed vocabulary tailored for BERT which consists of about 30 000 word and word piece [9] tokens. Words which are not in the vocabulary can be represented as multi-token words with the help of word piece tokens. Since the vocabulary includes even single letters, every possible input word has a valid tokenization. The input sentence is allowed to have a maximum length of 512 tokens. Second, the tokens are mapped to word embeddings, i.e. vectors $\mathbf{u}_i \in \mathbb{R}^H$, which represent the contextual meaning of each token. These vectors $\mathbf{u}_i$ are then fed into the first of twelve encoders. Each encoder consists of a multi-headed self-attention mechanism and a feed forward layer (i.e. multi-layer perception (MLP)). For a more detailed explanation of the inner workings of BERT, incl. normalization layers, positional encoding etc., the reader is referred to [7].

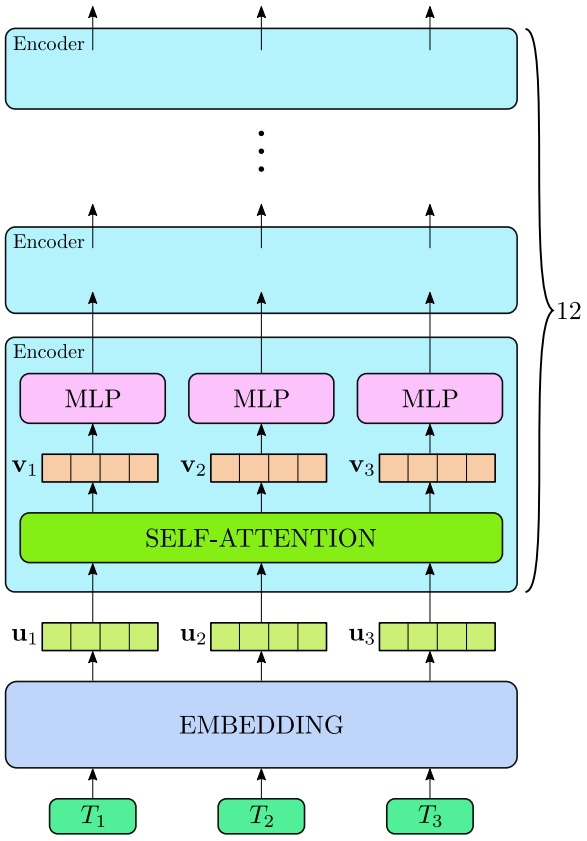

Figure 1. Simplified, high-level view of the BERT architecture [7, 8]. The word tokens are embedded and fed through twelve encoder blocks, each consisting mainly of a self-attention layer and a feed forward layer.

Since its release in 2019, BERT has gained great attention in the NLP community. There are two main reasons why BERT is chosen as a model for the conducted study: First, it shows superiority in performance compared to other model architectures in several NLP benchmark problems. It is particularly inspiring for this study that it outperforms the state-of-the-art NER models at the time

of its release. [8] Second, a large dataset is not necessarily needed in order to make use of BERT, since it is meant to be a model for transfer learning: pre-trained BERT models are already available for usage. They have been trained, in an unsupervised manner, with fake tasks such as next sentence prediction or masked language modeling. Large, plain-text corpora like Wikipedia or text books serve as a data source in this regard. These pre-trained language models require few additional training data if the task is only to fine-tune them with regard to a certain domain-specific language.

2.2 Industry Foundation Classes

Industry Foundation Classes (IFC) are a semantic standard [10] issued by the buildingSMART organization to aid in describing and exchanging building information in the AEC domain. Its purpose is to provide an interface for sharing data between different software tools. The IFC schema includes definitions of many construction related entities, as well as their properties and relations. These entities can be of varying nature: some describe physical parts of a building, e.g. IfcElements or IfcSpatialStructureElements, some describe more abstract things, e.g. IfcProcess, or even persons (IfcActor). An excerpt of the most general entities is shown in Figure 2.

3 Methodology

3.1 Data Preparation

In the conducted study, a text corpus is assembled from the public construction law texts of four of the 16 federated states in Germany. The documents are converted from PDF to plain text. Noise, such as page numbers, navigation links, etc., are removed manually where necessary. To subdivide the text according to sentence boundaries, the spaCy [12] sentence segmentation tool is used, which is based on sentence parsing. Since structural text elements such as headings or incomplete sentences do not carry any meaning, sentences with a word count below 15, including punctuation and special characters, are disregarded. All remaining clauses are full sentences and can be properly processed by the BERT model. The cleaned documents are annotated using the web-based annotation software INCEpTION [13].

3.2 Classification Scheme

The most typical entities to be detected by NER methods are generally terms like dates, documents, geopolitical entities (GPEs) and numerical values. For this study, in addition, some domain-specific entity labels are defined as entities in accordance with IFC classes. With the purpose of providing a proof of concept, only some high-level classes are selected, e.g. IfcActor, IfcBuilding, IfcBuildingElement and IfcSite, since they occur most frequently in the text. Table 1 lists all of the selected class labels and gives a brief description of what is included in each class.

3.3 Preprocessing

Before a tokenized sentence is fed to the model, it requires adjustment to meet the expected input format. Unlike RNN-based models, BERT expects an input of fixed size which consists of exactly 512 tokens. Therefore, each sentence is tokenized with the specialized BERT tokenizer and surrounded by the special tokens `[CLS]` (indicates sentence beginning) and `[SEP]` (indicates sentence ending). Sentences exceeding the maximum input length of 512 tokens are truncated, shorter sentences are padded with `[PAD]` tokens.
The experiments are carried out using three different tagging schemes, two of which require further pre-processing of the tokenized sentence. Simple, token-wise tagging of entities does not require further pre-processing, but it generally leads to a classifier with a vague understanding of where multi-token words start and end. To combat this impreciseness, the tagging scheme can be enriched by the IOB (I = inside, O = other, B = beginning) or BILOU (L = last, U = unit/single token) label-prefixes. [14] They can be derived directly from the already tokenized sentence by simple logic rules.

3.4 Model Architecture

The output matrix of BERT is of size $L \times H$, with sequence length L and hidden dimension H, i.e. it consists of one vector $\mathbf{z}_i \in \mathbb{R}^H$ for each input token T_i, with $0 \leq i < L$. A dropout layer with a dropout probability of $P_D = 0.3$ serves as a regularization measure, and is placed directly after the BERT model [15]. Each vector is fed forward through a single dense layer with output dimension N_C, which is the number of distinct classes. Note that for each position in the sentence, the weights of these dense layers are equal and thus they do not count as separate parameters. Thus, it only introduces 9216 additional weights to the model, if $N_C = 12$, i.e. it hardly increases model size. To predict the label for each token T_i, the arg max function that follows the dense layer selects the label with the highest score, per vector $\mathbf{x}_i \in \mathbb{R}^{N_C}$.
Since the corpus at hand consists solely of German text, the GBERT model presented in [16] is used, which has been pre-trained on various German corpora amounting to more than 160 GB of text data. The model architecture itself is identical to the one proposed in the original BERT

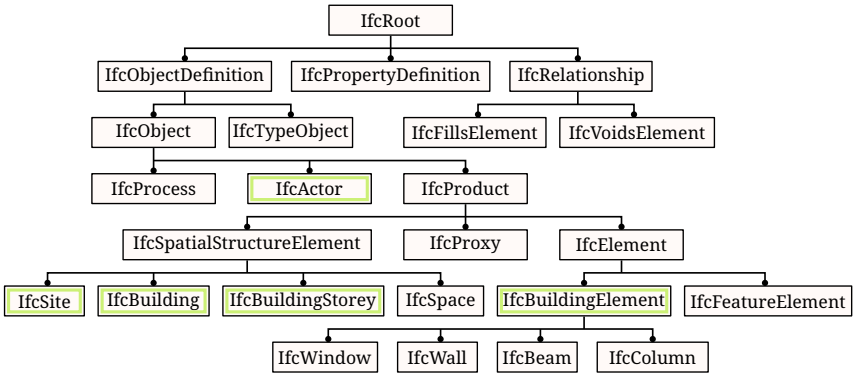

Figure 2. Partial view of the highest hierarchy levels of the IFC data model [11]. The highlighted classes are used for classification within this study.

Table 1. Performance scores of the model in the cross-validation experiment: model trained on all texts but one, and tested on the unseen model.

Class label	Key	Description, examples
IfcActor	ACT	organization, companies, persons
IfcBuilding	BUI	building types, e.g. schools, offices, or building definitions
IfcBuildingElement	BUE	building elements, such as doors, walls or windows
IfcBuildingStorey	STO	building story specification
IfcDate	DAT	temporal dates like "12.4.2013"
IfcSite	SIT	site of construction, dismantling etc.
IfcTransportElement	TRA	elevators, escalators, etc.
Document	DOC	certificates, assessments, official notices, etc.
Geopolitical Entity	GPE	countries, states, city states, etc.
Law Text	LAW	names of referenced law texts
Law Reference	REF	specifications of section/paragraph/clause in law text, e.g. "§ 70 Abs. 3"
Numerical Value	NUM	numerical value, possibly with unit, e.g. "7.5 m"

paper [8]. To investigate the influence of sheer model size, both $\text{GBERT}_{\text{BASE}}$ and $\text{GBERT}_{\text{LARGE}}$ are used as part of the model architecture in separate experiments and the test results are compared. This larger model's architecture is similar in concept, but more generous in terms of parameter count: it has 24 encoder blocks instead of 12, 16 attention heads per multi-headed attention block instead of 12, and has a hidden dimension of 1024.

4 Experimental Setup

To evaluate the performance of the proposed model, two experiments are conducted for this study:

1. **4-fold cross-validation**: To train a model on as much data as possible without compromising the amount of available test data, the model is N_D-fold cross-validated, where N_D is the number of documents. The sentence samples of $N_D - 1$ documents will be split up to parts of 90% training data and 10% validation data, while the sentences of the remaining document serve as test data.
2. **Training with little data**: To demonstrate the model's performance in the case that only few sentences are annotated, it is trained on only one input text at a time and tested on all others. This emulates the situation of domain experts, who cannot spend many hours with the annotation of building design rules, but instead only annotate a small portion of the corpus. Each training document is split up in the ratio of 9:1 to form training and validation sets, respectively. The rest of the corpus serves as a test set.

For both experiments, all three tagging schemes mentioned in Section 3.3 are tested and the results are compared.

4.1 Error Measure

Assigning labels to tokens within a sequence of text is a multi-class classification problem for each of the tokens. As a suitable loss function, the cross entropy loss is selected. It is computed for each of the tokens T_i and summed up over the whole sentence. The model's outputs at those positions which have special tokens (`[CLS]`,

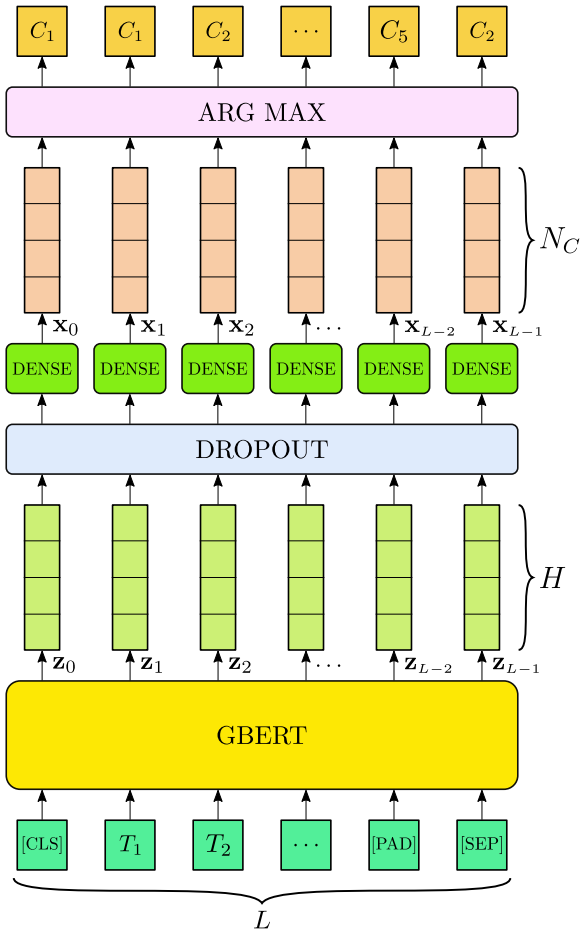

Figure 3. Schematic illustration of the named entity recognition model architecture used in this study. The arg max layer is used for predicting the most likely label, while the vectors $\mathbf{x}_i \in \mathbb{R}^{N_C}, 0 \leq i < L$ are used for loss computation.

[SEP] and [PAD]) as their input are not considered for loss computation.

Since only a small portion of words in the dataset are considered key terms, the number of unlabeled words is much higher than the number of actual named entities to detect. But even the classes themselves vary in size, which could lead to a biased classifier that favors more frequent class labels, especially the *no-entity* class. As addressed in [17], this class imbalance is a common problem in NLP and quite typical for NER tasks. As a remedy, a weighted cross entropy loss function is selected, and the loss is averaged with respect to each batch. For a given true class label C and the vector of logits $\mathbf{x} \in \mathbb{R}^{N_C}$, as predicted by the model, the loss is computed by

$$l_{CE}\left(\mathbf{x}, C\right) = w_C \left(-x_C + \ln\left(\sum_j e^{x_j}\right)\right) \tag{1}$$

where $\mathbf{w} \in \mathbb{R}^{N_C}$ is the vector of weight values assigned to each class label C. Since this loss function incorporates the negative log likelihood loss with the log softmax function, it expects raw output values from the model. Therefore, as shown in Figure 3, the model itself does not include any activation function. [18]

To calculate the weights w_C, the class label distribution in the respective training set is determined. Knowing the number of target labels occurrences per class t_C, the weights are computed by

$$\hat{w_C} = \sqrt{\frac{1}{t_C}} \tag{2}$$

and normalized by

$$w_C = \frac{\hat{w_C}}{\sum_{C=0}^{N_C} \hat{w_C}}. \tag{3}$$

This strategy introduces a higher penalty for the model whenever it predicts false-positives in the majority classes and a lower penalty for the model whenever it predicts false-positives in the minority classes, making up for the bias induced by the class-imbalance.

4.2 Learning schedule

In all training procedures of the described experiments, a constant learning rate of $l_r = 3 \times 10^{-6}$ is chosen. The batch sizes are $b_{\text{train}} = 16, b_{\text{val}} = 16, b_{\text{test}} = 16$ for training, validation and testing, respectively.

The number of epochs has the upper limit of $E_{\text{max}} = 100$, however, to prevent overfitting, it advisable to follow an early-stopping strategy. In this regard, a good compromise between minimizing both the out-of-sample error and the training time is to stop training after three consecutive epochs with an increase in validation error. [19] The model with the best validation error up to this point is then selected as the trained model. This robust and efficient mechanism is used in all experiments.

5 Results

To account for the class label imbalance, all precision and recall values presented in this section, and consequently all F1 scores, are weighted with regard to class label occurrences. [20] Note that this can lead to an F1 score which is not between precision and recall.

Table 2 summarizes the results of experiment 1. An averaged F1 score of 95.4% indicates the trained model's capability to be used for the task at hand. Even the lowest performance, if trained on a corpus of roughly 1900 sentences, was recorded to be 95.0% precision and 94.4%

Table 2. Performance scores of the model in the cross-validation experiment: model trained on all texts but one, and tested on the unseen text. Left column: ISO codes for the states of Germany whose building codes are included in the study.

Tested on	Precision	Recall	F1 score	Epochs
BE	0.958	0.953	0.955	18
BW	0.958	0.953	0.955	26
HB	0.950	0.944	0.946	36
HH	0.964	0.960	0.961	29
avg.	0.957	0.952	0.954	27

recall. Analogously, Table 3 shows the results of experiment 2. In comparison to experiment 1, the performance scores are, on average, slightly lower. This is plausible, as per training/test run, the model was trained on only one document instead of three. Nonetheless, even with such little training data, the model achieved averaged performance values of 93.5% precision and 92.6% recall.

In order to give a better insight into the incorrectly classified tokens, Table 5 displays the confusion matrix, created by accumulation of the confusion matrices of all token-wise labeling experiments listed in Table 2.

As it is of vital importance for the classifier to accurately detect entity beginnings and endings, its performance is examined for multiple tagging schemes. Table 4 shows some performance indicators obtained by experiment. As expected, the algorithm works best for the easiest of the addressed NER problems, i.e. without any tagging scheme. However, the results differ hardly if enhanced tagging schemes are used, as the average F1 score is 94.3% when using the IOB-scheme and also when using the BILOU-scheme. A greater discrepancy is the number of needed training epochs to achieve the stated performance scores: while training without any scheme is stopped, on average, after 27 epochs, it takes 31 epochs to train a model for the IOB scheme and 64 epochs for the BILOU scheme. In general, the learning curves are similar to the one displayed in Figure 5, which suggests that training for a few epochs might suffice to achieve almost saturated performance scores.

When using $\text{BERT}_{\text{LARGE}}$, the achieved F1 score is, on average, 0.2 percentage points lower in comparison to the results obtained with $\text{BERT}_{\text{BASE}}$. This shows, at least in this case and with the specified hyperparameters, the larger model is not worth the additional computational cost.

To give an idea of the produced output, an exemplary sentence with the detected entity labels is shown in Figure 4.

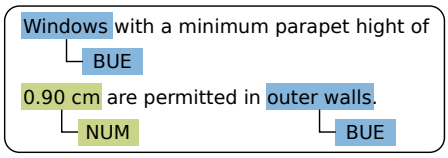

Figure 4. Example of analyzed sentence incl. the detected entity labels. Original sentence prior to translation: "Fenster sind in diesen Außenwänden ab einer Brüstungshöhe von 0,90 m zulässig".

Table 3. Performance scores and of the model trained on only one text and tested on all others.

Trained on	Precision	Recall	F1 score	Epochs
BE	0.919	0.910	0.913	34
BW	0.949	0.943	0.944	29
HB	0.938	0.927	0.930	41
HH	0.934	0.925	0.928	21
avg.	0.935	0.926	0.929	31

6 Conclusions

In this paper, a model architecture for NER mainly based on the pre-trained GBERT model is presented. When trained and tested on German public building code documents, average performance values of up to 95.7% precision and 95.2% recall are observed.

The main aspects of the conducted study are the following:

1. To the authors' knowledge, this is the first application of a transformer-based deep learning model to German text in the construction domain. The feasibility of the concept is proven and further development of processes based on the presented NER method is made possible.

2. In the course of the study, a data set is created which contains about 2500 sentences from German construction law documents. The clauses have been cleaned, processed for usage and have been manually annotated.

It is illustrated that the proposed model functions well, even with small amounts of training data. This makes it possible to use the model, with little annotation effort, as a building block in other processes. Possible applications could help with the automatic semantic enrichment of

Table 4. Average performance scores of the model with regard to the used tagging scheme.

Tagging scheme	Prec.	Recall	F1 score	Epochs
Experiment 1:				
token-wise	0.958	0.953	0.954	27
IOB	0.949	0.941	0.943	31
BILOU	0.950	0.941	0.943	64
Experiment 2:				
token-wise	0.935	0.927	0.929	31
IOB	0.923	0.910	0.913	41
BILOU	0.932	0.923	0.924	85

Table 5. Accumulated confusion matrix of all the processed test sets. The left column shows the true labels, the upper row shows the predicted labels.

True \ Pred.	DAT	DOC	GPE	ACT	BUI	BUE	SIT	STO	TRA	LAW	LAT	NUM	O
DAT	2275	1	0	0	1	1	1	1	0	10	11	7	13
DOC	1	2001	0	17	3	0	0	0	0	4	45	0	91
GPE	0	0	121	4	0	0	0	0	0	0	6	0	0
ACT	4	23	5	2927	29	3	4	2	0	2	1	0	86
BUI	4	1	0	23	3344	95	12	1	2	41	3	11	300
BUE	0	0	0	3	176	2612	4	18	3	0	2	4	307
SIT	0	3	0	0	2	2	317	0	0	0	0	0	10
STO	0	0	0	0	2	0	0	251	0	0	0	4	5
TRA	0	0	0	0	8	0	0	0	58	0	0	0	2
LAW	18	1	0	2	7	2	0	2	0	4076	39	1	41
LAT	18	8	14	3	1	0	0	0	0	13	1108	8	86
NUM	0	0	0	1	2	2	0	6	0	14	0	1444	20
O	36	619	25	566	962	875	158	81	27	76	212	58	47782

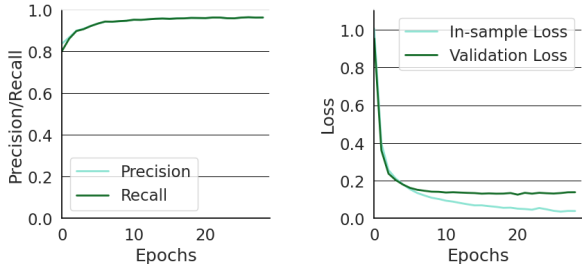

Figure 5. Exemplary training curves for cross-validation experiment 1, with BILOU tagging scheme. Left: performance scores obtained from applying the model to the test set after each epoch. The precision and recall curves are almost identical. Right: learning curve, in- and out-of sample loss progression.

BIM models or with automatic rule extraction from legal documents in the construction domain.

A possible limitation of the method is that it might only work well within the given corpus of law texts, which are admittedly quite similar to each other. One could argue that the technical and legal terms in the construction codes are, by their nature, quite repetitive and that it is no surprise they are detected, since then the model basically acts as a string matcher. Therefore it remains to be examined if the model would indeed outperform such a rule-based approach.

Future works should include, first, a more fine-grained categorization of entities to fit more and also more specific entities in the IFC schema. This is necessary to produce an exhaustive mapping from terms in law clauses to IFC objects. Second, to extract information regarding the relations between those objects, the dataset needs to be extended by another layer of annotations and a second model has to be introduced to detect these relations. Third, a larger dataset should be created that contains all of the 16 federated state building codes, and also other types of legal documents to examine the generalization of the trained model.

References

[1] Fulin Li, Yuanbin Song, and Yongwei Shan. Joint Extraction of Multiple Relations and Entities from Building Code Clauses. *Applied Sciences*, 10(20): 7103, 2020. doi:10.3390/app10207103.

[2] Ruichuan Zhang and Nora El-Gohary. A Machine-Learning Approach for Semantic Matching of Building Codes and Building Information Models (BIMs) for Supporting Automated Code Checking. In Hugo Rodrigues, George Morcous, and Mohamed Shehata, editors, *Recent Research in Sustainable Structures*, Sustainable Civil Infrastructures, pages 64–73, Cham, 2020. Springer International Publishing. doi:10.1007/978-3-030-34216-6_5.

[3] Seonghyeon Moon, Gitaek Lee, Seokho Chi, and Hyunchul Oh. Automated Construction Specification Review with Named Entity Recognition Using Natural Language Processing. *Journal of Construction Engineering and Management*, 147(1), 2021. doi:10.1061/(ASCE)CO.1943-7862.0001953.

[4] Jaeyeol Song, Jin-Kook Lee, Jungsik Choi, and Inhan Kim. Deep learning-based extraction of predicate-argument structure (PAS) in building design rule sentences. *Journal of Computational Design and Engineering*, 7(5):563–576, 2020. doi:10.1093/jcde/qwaa046.

[5] Botao Zhong, Xuejiao Xing, Hanbin Luo, Qirui Zhou, Heng Li, Timothy Rose, and Weili Fang. Deep learning-based extraction of construction procedural constraints from construction regulations. *Advanced Engineering Informatics*, 43:101003, 2020. doi:10.1016/j.aei.2019.101003.

[6] Weili Fang, Hanbin Luo, Shuangjie Xu, Peter E. D. Love, Zhenchuan Lu, and Cheng Ye. Automated text classification of near-misses from safety reports: An improved deep learning approach. *Advanced Engineering Informatics*, 44:101060, 2020. doi:10.1016/j.aei.2020.101060.

[7] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 6000–6010, Red Hook, 2017. Curran Associates Inc. URL https://dl.acm.org/doi/10.5555/3295222.3295349.

[8] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, 2019. Association for Computational Linguistics. doi:10.18653/v1/N19-1423.

[9] Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural Machine Translation of Rare Words with Subword Units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, 2016. Association for Computational Linguistics. doi:10.18653/v1/P16-1162.

[10] *Industry Foundation Classes (IFC) for data sharing in the construction and facility management industries - Part 1: Data schema*. ISO 16739-1:2018-11. 2018. doi:10.31030/2584995.

[11] André Borrmann, Markus König, Christian Koch, and Jakob Beetz, editors. *Building Information Modeling: Technologische Grundlagen und industrielle Praxis*. Springer Vieweg, Wiesbaden, 2015. doi:10.1007/978-3-658-05606-3.

[12] Matthew Honnibal, Ines Montani, Sofie Van Landeghem, and Adriane Boyd. spaCy: Industrial-strength Natural Language Processing in Python, 2020. URL https://doi.org/10.5281/zenodo.1212303.

[13] Jan-Christoph Klie, Michael Bugert, Beto Boullosa, Richard Eckart de Castilho, and Iryna Gurevych. The INCEpTION Platform: Machine-Assisted and Knowledge-Oriented Interactive Annotation. In *Proceedings of the 27th International Conference on Computational Linguistics: System Demonstrations*, pages 5–9, Santa Fe, 2018. Association for Computational Linguistics. URL https://aclanthology.org/C18-2002/.

[14] Jing Li, Aixin Sun, Jianglei Han, and Chenliang Li. A Survey on Deep Learning for Named Entity Recognition. *IEEE Transactions on Knowledge and Data Engineering*, pages 1–1, 2020. doi:10.1109/TKDE.2020.2981314.

[15] Geoffrey E. Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R. Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv:1207.0580 [cs]*, 2012. URL http://arxiv.org/abs/1207.0580.

[16] Branden Chan, Stefan Schweter, and Timo Möller. German's Next Language Model. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 6788–6796, Barcelona (Online), 2020. International Committee on Computational Linguistics. doi:10.18653/v1/2020.coling-main.598.

[17] Xiaoya Li, Xiaofei Sun, Yuxian Meng, Junjun Liang, Fei Wu, and Jiwei Li. Dice Loss for Data-imbalanced NLP Tasks. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 465–476, Online, 2020. Association for Computational Linguistics. doi:10.18653/v1/2020.acl-main.45.

[18] Adam Paszke et al. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d' Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32, pages 8024–8035. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper/2019/file/bdbca288fee7f92f2bfa9f7012727740-Paper.pdf.

[19] Lutz Prechelt. Early Stopping - But When? In Genevieve B. Orr and Klaus-Robert Müller, editors, *Neural Networks: Tricks of the Trade*, Lecture Notes in Computer Science, pages 55–69. Springer, Berlin, Heidelberg, 1998. doi:10.1007/3-540-49430-8_3.

[20] Thong Nguyen, Duy Nguyen, and Pramod Rao. Adaptive name entity recognition under highly unbalanced data. *CoRR*, abs/2003.10296, 2020. URL https://arxiv.org/abs/2003.10296.