Logical Structure-based Pretrained Models for Legal Text Processing

Nguyen Ha Thanh[a] and Nguyen Le Minh[b]
Japan Advanced Institute of Science and Technology, Japan

Keywords: Logical Structure, Law, Pretrained Model.

Abstract: In recent years, we have witnessed breakthroughs in natural language processing coming from pretrained models based on the Transformer architecture. In the field of legal text processing, a special sub-domain of NLP, pretrained models also show promising results. For a legal sentence, although the natural language is used for expression, the real meaning lies in its logical structure. From that observation, we have a hypothesis that the knowledge of recognizing logical structures can support deep learning models to understand the legal text better and achieve a higher performance in the related tasks. To verify our assumption, we design a novel framework to inject the knowledge about recognizing the requisite and effectuation part of a law sentence into Transformer models. Our proposed method is effective and general. By our experiments, we provide informative results about our approach and its performance compared with the baselines.

1 INTRODUCTION

Deep learning models find common abstract patterns in data and use them to predict similar patterns in the future. For that reason, the quantity and quality of the data affect the model's performance. In practice, the data is not always in abundance for the model to learn such information on its own. Hence, the transfer learning and knowledge injection methods (Nguyen et al., 2020a; Lauscher et al., 2020; Wang et al., 2019; Shafahi et al., 2019) become the best practice for deep learning models in data-deficient domains.

Law processing is a complex area of NLP. This complexity comes from the very language of law (Nguyen and Nguyen, 2021) and the requirement of the problems in this field. In order for AI systems to be truly useful in the legal domain, their requirements are often much higher than business applications like sentiment analysis or recommendation systems. The language of law is often not easy to understand to lay readers. Besides, the data to train these models is not always available, therefore it is challenging for the model to abstract out useful patterns in an unsupervised training manner. Hence, an appropriate knowledge injection method can improve the performance in legal text processing using deep learning models.

Logic structures are integral parts of legal sentences. Identifying criminals, breaches of contracts, and a host of other important legal decisions are all based on logic. A novel author can use better language than a lawyer but standing in court they cannot justify a person from the death penalty with their words without logic. Similarly, a language model trained on a giant corpus without knowledge of logic is intrinsically useless in law. This can make it difficult to answer the inference questions of the law, which are critical in being able to bring the results to reality.

In an experiment for Transformer models solving math problems (Saxton et al., 2019), these models have amazing results with simple addition and subtraction. However, with the combined operations, the performance is reduced by 50%. This shows that the model uses only interpolations to predict the outcome than to actually calculate it logically. Mathematics is formed by operations, law sentences are formed by logical structures and it is a fragile belief that these models can learn the logical structures in the sentences by itself. There needs to be a more effective way to instruct the model to obtain abstractions for logical structures.

From the observations above, in this paper, we introduce TRE framework which is a knowledge injection framework for Transformer based models with requisite and effectuation data. Applying this framework, we investigate and pretrain variants of TRE-BERT, pretrained models on BERT. Our experiments prove the effectiveness of this approach. The three

[a] https://orcid.org/0000-0003-2794-7010
[b] https://orcid.org/0000-0002-2265-1010

Nguyen, H. and Nguyen, L.
Logical Structure-based Pretrained Models for Legal Text Processing.
DOI: 10.5220/0010852000003116
In *Proceedings of the 14th International Conference on Agents and Artificial Intelligence (ICAART 2022) - Volume 3*, pages 524-531
ISBN: 978-989-758-547-0; ISSN: 2184-433X

main contributions of this paper include:

- Propose TRE framework, a novel knowledge injection pretraining framework for Transformer models;
- Introduce TREBERT pretrained with special hierarchical logical structures to improve performance of BERT in the legal domain;
- Experiment in detail to prove the effectiveness as well as understand the characteristic of the proposed approach.

2 RELATED WORK

Methods using pretrained models have proven effective in a variety of problems in natural language processing (Qiu et al., 2020). In legal text processing, a subfield of NLP, there are also studies that apply this approach and achieve good results. There are two ways to pretrain these models. The first one is training from scratch, in which, initialization of tokenizer, initialization of weight is done from scratch. This approach often requires large amounts of data so that the models can abstract the patterns on their own. The second one is further pretraining, where the pretrained models will continue to be trained to better abstract the relationships between concepts in the domain. Pretraining studies in law evolved with the development of the architecture of deep learning.

Law2Vec (Chalkidis and Kampas, 2019) is trained on a large vocabulary from UK, EU, Canada, Australia, USA, and Japan law data. At that time, the authors assumed that the resource on word embedding for the field of law was limited, researchers often had to use general-purpose word embeddings in the problem of law, which resulted in models not achieving their full potential. The authors demonstrate that using word vectors trained from a large corpus of law results in better performance for deep learning models. The idea of approaches using pretrained word vectors is that it is feasible to measure the co-occurrence of terms in a corpus. In terms of pretraining a model for legal text processing, a broad corpus is better than a narrow corpus, corpus with more legal terms is better than a corpus in the general domain.

Besides methods using word vectors, methods using pretrained contexture embedding with Transformer architecture are also competitive approaches in the field of law. The authors of Legal BERT(Chalkidis et al., 2020) create variants in the legal domain for BERT (Devlin et al., 2018), examining both pretraining from scratch and further pretraining. Through experiment, the authors show that both approaches have better performance in legal text processing than using the original model pretrained in the general domain. In the task of recognizing the named entity for the contract, the version pretrained from scratch outperformed the further pretrained version. At the same time, JNLP Team (Nguyen et al., 2020b) propose two systems using BERT pretrained from scratch and further pretrained, which become the best systems in case law entailment and statute law question answering tasks in COLIEE 2020 (Rabelo et al., 2020).

Analyzing the previous pretraining methods in the legal domain, we find that these methods have the same thing in common, that they are pretrained unsupervised on a large corpus. By doing so, we can create language models that accurately describe the relationships of concepts, terms, and syntax used in legal documents. These models can also find hidden rules expressed in words, use extrapolation to make decisions. However, it is impossible for the model to find all the latent rules just by identifying co-occurring terms. This is a process that requires much time, a lot of computational power, a huge amount of data. Similar to the issues in math problems, it is difficult for the model to find logical rules through unsupervised training. Our approach is a further pretraining method based on a supervised paradigm.

3 METHOD

3.1 Legal Logical Structures

With a different purpose than daily life sentences, legal sentences often require rigor and logic. As the product of thousands of years of human civilization, the logic of the existing laws reaches a very high level. From a syntactic point of view, two important components of a law sentence to form an equivalent logical proposition are requisite and effectuation. Requisite and effectuation can be formed from smaller logical parts such as antecedence, consequence, and topic. In their work, (Bach et al., 2010) indicate four different cases of these structures.

With a classic example in the logic “If it rains, the road is wet.”, we can easily see that this sentence has a requisite segment and an effectuation segment. The requisite and effectuation segments are often complex in practice, they can be nested and even interleaved. In legal sentences, besides *requisite* and *effectuation*, another common logical structure is *unless*, which indicates exceptions where the main requisite and effectuation do not apply. Let’s consider the following example: “Gifts not in writing may be revoked by ei-

ther party; provided, however, that this shall not apply to any portion of the gift for which performance has been completed." With such a complex sentence, it is easy to see that there is more than one requisite and effectuation pair in this sentence. Therefore, it is difficult for a language model with averaging and interpolation capabilities to infer logical structures on its own through unsupervised training. To correctly annotate law sentences with many interlocking logical structures, we need to use multilayer annotation (Nguyen et al., 2018). Table 1 is the annotation of the above example.

As with the math problem, if the model cannot separate the logical structures and still answer the questions correctly, there are two possibilities. Either the model answered correctly by luck or because of the same bias that occurs between the training and inference phases. In either case, the model doesn't really understand the meaning of the sentence and so it cannot meet the requirements of real applications. Therefore, one logical idea that can be suggested here is to use knowledge of the requisite and effectuation recognition in legal sentences to guide the model towards the right abstraction of logical structures.

3.2 TRE Framework

With the goal of building a Transformer model that can learn to recognize the segments of logical structures, we propose the TRE (Transferred-Requisite-Effective) Framework. This framework makes it possible to inject the logical structure information into the self-attention layers of the Transformer so that the model can form the corresponding abstractions. Unlike conventional pretraining approaches, labels are usually provided at the last Transformer layer, within this framework, information about logical structures can be injected into the hidden layers (Figure 1).

With this framework, knowledge injection is done through a gradient descent process. Instead of rigidly specifying information about logical structures through constants, the model's parameters need to be updated so that corresponding abstractions can be formed. The novelty of this framework lies in the Transferred-Requisite-Effective self-attention layers (TRE layers). These layers stores the knowledge about recognizing logical structures in the legal sentence, which is learned from the provided labels in pretraining data.

Suppose after the layer $i-1^{th}$, we have a signal sequence of length M which can be presented as $E^{i-1} = (e_1^{i-1}, e_2^{i-1}, ..., e_M^{i-1})$. The i^{th} layer is a TRE layer, basically, the information flow is the same as in a regular Transformer layer. At the i^{th} layer, vector E^{i-1} is multiplied by the attention matrices Q^i, K^i, V^i to get the corresponding attention vectors. These attention vectors are combined according to Equation 1 to get the corresponding output at the i^{th} transfomer layer $Z^i = (z_1^i, z_2^i, ..., z_M^i)$.

$$Z^i = softmax(\frac{Q^i \times K^{i\top}}{\sqrt{d}} V^i) \quad (1)$$

In the forward direction of Transformer architecture, we normalize Z^i with a layer normalization (Ba et al., 2016) and a dense layer, the signal can also be transmitted directly through the residual connections. To pass the labels of the logical structures to the network, at the branching direction as an injection needle, Z^i after going through a dense layer and the softmax function as in Equation 2, predicted labels and gold labels are compared and the loss is backpropagated for updating parameters.

$$L = softmax(Z^i \times W + b) \quad (2)$$

Theoretically, the TRE layers could be in any section of the pipeline. Through experiments, we can find the optimal position of TRE layers for given data.

3.3 TREBERT

TRE Framework is a general idea for models with Transformer architecture with self-attention layers. From that general idea, within the confines of this paper, we pretrain a popular variant of Transformers, BERT (Devlin et al., 2018), analyze and verify the performance. Implementation for other variants can be implemented in future works.

In the original version and variants of BERT, the model was pretrained with unsupervised tasks, the most common being Masked Language Modeling and Next Sentence Prediction. With the TRE Framework, TREBERT is trained with the task of hierarchical logical structures recognition. For the same input law sentence, at injection needles, we pass the labels corresponding to the hierarchical layer of the interlocking logical structures which the TRE layer is responsible for.

We use JCC-RRE dataset (Nguyen et al., 2018) to pretrain the model. Data is tagged according to the BIOE schema for each logical structure Requisite, Effectuation, and Unless. Figure 2 shows the distribution of logical structures and Table 2 shows the distribution of tags in this dataset. Examples in this dataset are annotated in 3 layers, the overlapping logical structures are fully represented in the data. We pretrain TREBERT under the paradigm of multitasking learning on TRE layers. Each TRE layer is responsible for one logical layer. We experiment to find the best configuration of TRE framework.

Table 1: Multilayer annotation in the BIOE schema of *requisite*, *effectuation* and *unless* segments for the sample "Gifts not in writing may be revoked by either party; provided, however, that this shall not apply to any portion of the gift for which performance has been completed."

Token	Layer 1	Layer2	Layer 3
Gifts	B-R	B-E	-
not	I-R	-	-
in	I-R	-	-
writing	E-R	-	-
may	-	I-E	-
be	-	I-E	-
revoked	-	I-E	-
by	-	I-E	-
either	-	I-E	-
party	-	E-E	-
;	-	-	-
provided	-	-	B-U
,	-	-	I-U
however	-	-	I-U
,	-	-	I-U
that	-	-	I-U
this	-	B-E	I-U

Token	Layer 1	Layer2	Layer 3
shall	-	I-E	I-U
not	-	I-E	I-U
apply	-	I-E	I-U
to	-	I-E	I-U
any	-	I-E	I-U
portion	B-R	I-E	I-U
of	I-R	I-E	I-U
the	I-R	I-E	I-U
gift	I-R	E-E	I-U
for	I-R	-	I-U
which	I-R	-	I-U
performance	I-R	-	I-U
has	I-R	-	I-U
been	I-R	-	I-U
completed	E-R	-	E-U
.	-	-	-

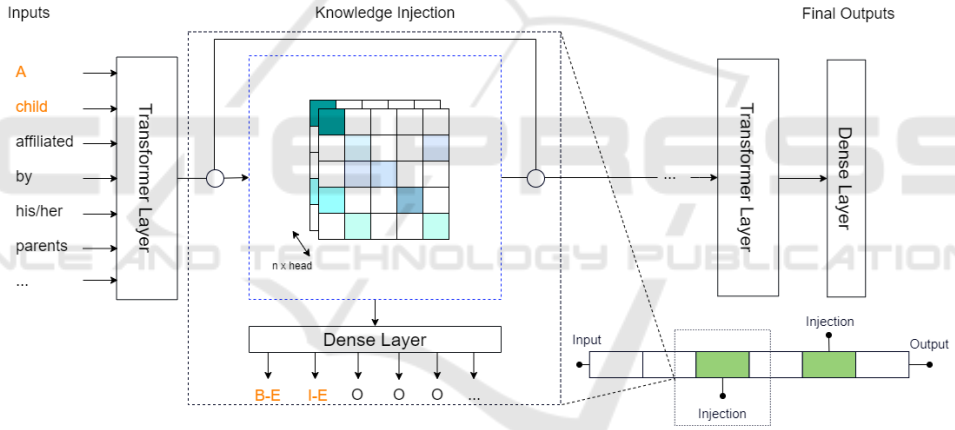

Figure 1: General flow of TRE Framework for Transformer models.

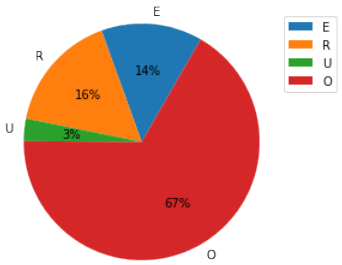

Figure 2: Logical structure distribution of the pretraining data (E = Effectuation, R = Requisite, U = Unless, O = Other).

Table 2: Tag distribution following BIOE schema of the pretraining data.

Tag	Meaning	Occurence
B-E	Begin Effectuation Part	1,982
B-R	Begin Requisite Part	2,408
B-U	Begin Unless Part	260
E-E	End Effectuation Part	2,088
E-R	End Requisite Part	2,395
E-U	End Unless Part	259
I-E	Inside Effectuation Part	26,762
I-R	Inside Requisite Part	31,474
I-U	Inside Unless Part	6,270
O	Others	148,540

4 PRETRAINING TREBERT

We pretrain TREBERT as described in Section 3. This pretraining process has the following characteristics. Firstly, it is further pretraining on a Transformer model, i.e. the original model needs to be pretrained on basic linguistic tasks to form contexture embedding on a specific vocabulary before being pretrained with logical structure data. Secondly, it is implemented as supervised learning, i.e. trained with labeled data. Thirdly, the TRE layers where the label is injected will be trained in parallel.

From the characteristics of this process, we see that, instead of assigning just one configuration, we can experiment with different configurations of the framework. This experiment not only helps to find the best configuration but also helps us better understand the behavior of the model during this phase. Since pretraining is done in the form of supervised training, we can track the performance of the model on a validation set. Our assumption is that a good configuration can help the model to make a good abstraction, thereby making accurate predictions.

With statistics from Table 2, we can see that this is a classification problem with unbalanced labels. Therefore, we use the precision, recall, and F1 metrics on logical structure parts to evaluate and compare configurations, a correct logical structure part must have all correctly predicted labels.

We pretrain the variants of TREBERT considering two aspects: the position of the TRE Layers and the loss portion between them. Just considering the position of the TRE layer on the 12 Transformer layers of BERT, we have ${}^{12}P_3 = 1320$ cases. Testing all configurations is resource-intensive, so we use the boundary value analysis technique to generate representative configurations for the positions and random search for the loss portion. Table 3 represents positional configurations and their performances on pretraining data. We also included in the table 30 configurations where the TRE layers are randomly decided. For a multi-task training problem, the final loss function is calculated by the following equation:

$$\mathcal{L} = \alpha * \mathcal{L}_\alpha + \beta * \mathcal{L}_\beta + \gamma * \mathcal{L}_\gamma$$

In which, $\mathcal{L}$ be the total loss, $\mathcal{L}_\alpha$, $\mathcal{L}_\beta$ and $\mathcal{L}_\gamma$ be the individual loss in each injected position with the corresponding weights α, β and γ, which can be optimized via hyperparameter tuning techniques. When conducting experiments with TRE layer position, we fixed the portion loss of all three logical structure layers equally.

From the experiment, it can be seen that the good positions to inject the knowledge of the logical structure are in the deeper layers. This can be explained that the knowledge injection needs to match the level of abstraction of the neural network. For deep learning models, in the early layers, the abstractions are low-level. At the deeper layers, the level of abstraction increases. The knowledge injected into the early layers will deviate from the abstraction level compared to the unsupervised features formed during the previous pretraining. The table also shows that the distance between the injection needle positions should be 1 to 2 layers. We select the 3 best configurations to continue conducting random searches to find the best portion of the loss functions for each. Different positional configurations have different optimal loss portions.

5 EXPERIMENTS

5.1 Experimental Settings

We use question answering data of COLIEE 2021 competition to verify the effectiveness of TREBERT. For each statement, the model needs to predict whether the statement is lawful or not. This is data that can evaluate the strength of pretrained models because of the small number of samples. The systems need to perceive the semantics in the sentence to give the correct answer.

In the data provided by COLIEE, the official test set includes 81 yes/no questions. To generate the training data, we use the method proposed by (Nguyen et al., 2019), data from previous years and from the Japanese civil code are augmented with simple negation rules. After the augmentation process, we have 4000 samples. We spend 10% for the development set and the rest is for finetuning TREBERT and baseline models.

Our experimental baselines include original BERT, LegalBERT_SC (Legal BERT from scratch), and LegalBERT_FP (Legal BERT further pretrained). These baselines have the common feature of being pretrained unsupervised on lexical tasks and not pretrained with logical structures. The measurement used in this task is accuracy. TREBERT_8_10_12, TREBERT_7_8_9 and TREBERT_7_9_11, the variants of TREBERT that performed best on the pretraining task, are included in the experiment. In addition, we also used a configuration with poor pretraining results, TREBERT_1_2_3, to further understand the behavior of this model family.

To ensure fairness in the number of parameters, the baselines and variants of TREBERT all use the architecture and configuration of BERT Base Uncased. Note that TREBERT's injection needles are removed

Table 3: Representative positional configuration and their performances on the validation set. TREBERT_X_Y_Z stands for the best configuration that the knowledge is injected in Transformer layers X^{th}, Y^{th} and Z^{th}.

Configuration	Precision	Recall	F1 Score
Uniform loss portion			
TREBERT_8_10_12	0.5890	0.7000	0.6373
TREBERT_7_8_9	0.5441	0.7102	0.6151
TREBERT_7_9_11	0.5420	0.7305	0.6140
TREBERT_10_11_12	0.5544	0.6661	0.6064
TREBERT_6_9_12	0.5262	0.7424	0.5959
TREBERT_4_8_12	0.4500	0.6966	0.5102
TREBERT_2_7_12	0.3994	0.7136	0.4549
TREBERT_4_5_6	0.3550	0.5814	0.4235
TREBERT_1_6_11	0.3490	0.7085	0.4027
TREBERT_2_4_6	0.3266	0.6678	0.3894
TREBERT_1_5_9	0.3260	0.7254	0.3704
TREBERT_1_4_7	0.2745	0.6695	0.3254
TREBERT_1_3_5	0.2199	0.5695	0.2726
TREBERT_1_2_3	0.1655	0.4932	0.2124
Avg Random (30 runs)	0.3654	0.5826	0.4088
Optimized loss portion			
TREBERT_8_10_12	0.5807	0.7373	0.6555
TREBERT_7_8_9	0.5725	0.6814	0.6313
TREBERT_7_9_11	**0.6300**	**0.7723**	**0.6939**

Table 4: Performance on the test set.

Model / Configuration	Correct	Accuracy
TREBERT_8_10_12	52	0.6420
TREBERT_7_9_11	52	0.6420
TREBERT_7_8_9	49	0.6049
LEGAL_BERT_SC	47	0.5802
LEGAL_BERT_FC	46	0.5679
Original BERT	44	0.5432
TREBERT_1_2_3	44	0.5432

after pretraining so they don't affect the parameter count. The experiments are conducted with GPU Tesla P100-PCIE-16GB.

5.2 Experimental Results

Experimental results in Table 4 shows that the configurations TREBERT_8_10_12 and TREBERT_7_9_11 lead the rankings with 52 correct answers out of a total of 81 questions in the test set, followed by TREBERT_7_8_9, LEGAL_BERT_SC and LEGAL_BERT_FC with 49, 47 and 46 correct answers. Original BERT and TREBERT_1_2_3 answered 44 questions correctly. This result shows us that the hypothesis made at the beginning of the paper is reasonable. Pretraining the models using a logical structure helps them to make better predictions in tasks that require understanding in the legal domain. Compared with the results announced by COLIEE-2021[1], top TREBERT variants (TREBERT_8_10_12, TREBERT_7_9_11) achieved state-of-the-art performance.

To better understand the behavior of the model, we visualize the self-attention weight of the last layer where the injection needle is attached. If the model generates the correct abstraction, the attention matrix must reflect that. Figure 3 is about the self-attention visualization of the 11th layer of TREBERT_7_9_11 with the input as "Gifts not in writing may be revoked by either party; provided, however, that this shall not apply to any portion of the gift for which performance has been completed.".

Looking at Figure 3, it can be seen that TREBERT_7_9_11 has adjusted its attention to the logical structures. With the token "gift", from Table 1, we can see that it belongs to both logical parts requisite and effectuation. TREBERT_7_9_11's attention weights correctly tie "gift" to the tokens to the requisite (gift not in writing) and to the effectuation (gift may be revoked by either party) parts. This could explain the outperforming on the test set of the model.

TREBERT_1_2_3 has bad performance on the pretraining task, resulting in no performance improvement compared to the original BERT. As analyzed above, knowledge injection fails due to the incompatibility of abstraction between data and model archi-

[1]https://sites.ualberta.ca/ rabelo/COLIEE2021/results /task5_res.html

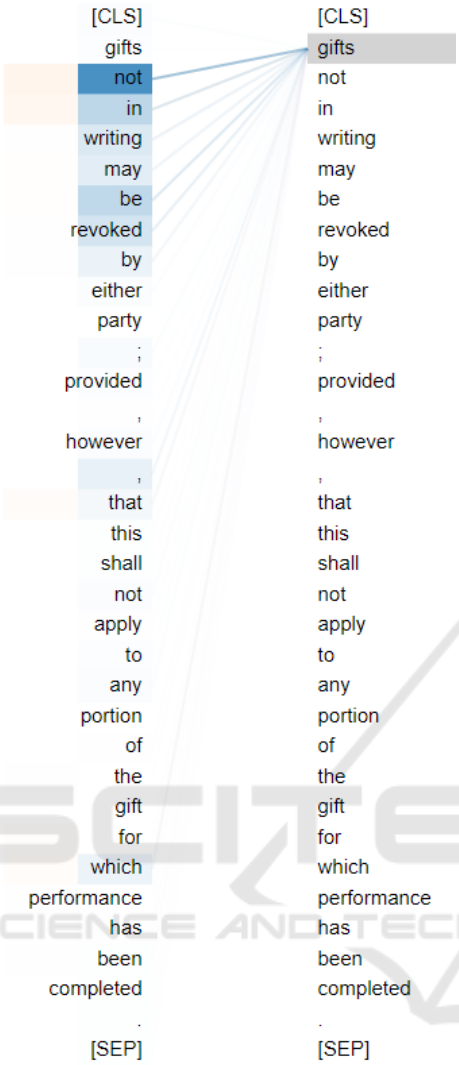

Figure 3: Self-attention visualization of the 11th layer of TREBERT_7_9_11 with the input as "Gifts not in writing may be revoked by either party; provided, however, that this shall not apply to any portion of the gift for which performance has been completed.".

tecture. This result is consistent with the assumption that a model capable of analyzing the logical structures can answer the legal questions better. Working with different configurations in the pretraining phase helps us find the appropriate positions to inject each type of knowledge.

6 CONCLUSIONS

In this study, we propose and investigate TRE framework, a knowledge injection approach for pretrained Transformer models. We then apply the TRE framework to pretrain the variants of TREBERT from the original BERT model. Our detailed experiments and surveys show the effectiveness and explainability of the method. A model having good skill in recognizing logical structure performs better on legal question answering.

Although in this paper the TRE framework is proposed with logical structures, the idea is general and extensible. Variations of knowledge can be experimented in different tasks. In future studies, we want to inject NLP annotated resources into Transformer models with TRE framework to get better and more explanatory pretrained models. In addition, the limitation of this approach is that finding good configurations consumes a lot of computing power for hyperparameter optimization, overcoming this limitation is also an interesting research direction.

ACKNOWLEDGEMENTS

This work was supported by JSPS Kakenhi Grant Number 20H04295, 20K2046, and 20K20625.

REFERENCES

Ba, J. L., Kiros, J. R., and Hinton, G. E. (2016). Layer normalization. *arXiv preprint arXiv:1607.06450*.

Bach, N. X., Le Minh, N., and Shimazu, A. (2010). Recognition of requisite part and effectuation part in law sentences. In *Proceedings of (ICCPOL)*, pages 29–34.

Chalkidis, I., Fergadiotis, M., Malakasiotis, P., Aletras, N., and Androutsopoulos, I. (2020). LEGAL-BERT: The muppets straight out of law school. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 2898–2904, Online. Association for Computational Linguistics.

Chalkidis, I. and Kampas, D. (2019). Deep learning in law: early adaptation and legal word embeddings trained on large corpora. *Artificial Intelligence and Law*, 27(2):171–198.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Lauscher, A., Majewska, O., Ribeiro, L. F., Gurevych, I., Rozanov, N., and Glavaš, G. (2020). Common sense or world knowledge? investigating adapter-based knowledge injection into pretrained transformers. *arXiv preprint arXiv:2005.11787*.

Nguyen, H., Tran, V., and Nguyen, L. (2019). A deep learning approach for statute law entailment task in coliee-2019. *Proceedings of the 6th Competition on Legal Information Extraction/Entailment. COLIEE*.

Nguyen, H.-T. and Nguyen, L.-M. (2021). Sublanguage:

A serious issue affects pretrained models in legal domain. *arXiv preprint arXiv:2104.07782*.

Nguyen, H. T., Vu, T. K., Racharak, T., Nguyen, L. M., and Tojo, S. (2020a). Knowledge injection to neural networks with progressive learning strategy. In *International Conference on Agents and Artificial Intelligence*, pages 280–290. Springer.

Nguyen, H.-T., Vuong, H.-Y. T., Nguyen, P. M., Dang, B. T., Bui, Q. M., Vu, S. T., Nguyen, C. M., Tran, V., Satoh, K., and Nguyen, M. L. (2020b). Jnlp team: Deep learning for legal processing in coliee 2020. *arXiv preprint arXiv:2011.08071*.

Nguyen, T.-S., Nguyen, L.-M., Tojo, S., Satoh, K., and Shimazu, A. (2018). Recurrent neural network-based models for recognizing requisite and effectuation parts in legal texts. *Artificial Intelligence and Law*, 26(2):169–199.

Qiu, X., Sun, T., Xu, Y., Shao, Y., Dai, N., and Huang, X. (2020). Pre-trained models for natural language processing: A survey. *Science China Technological Sciences*, pages 1–26.

Rabelo, J., Kim, M.-Y., Goebel, R., Yoshioka, M., Kano, Y., and Satoh, K. (2020). Coliee 2020: Methods for legal document retrieval and entailment.

Saxton, D., Grefenstette, E., Hill, F., and Kohli, P. (2019). Analysing mathematical reasoning abilities of neural models. *arXiv preprint arXiv:1904.01557*.

Shafahi, A., Saadatpanah, P., Zhu, C., Ghiasi, A., Studer, C., Jacobs, D., and Goldstein, T. (2019). Adversarially robust transfer learning. *arXiv preprint arXiv:1905.08232*.

Wang, K., Gao, X., Zhao, Y., Li, X., Dou, D., and Xu, C.-Z. (2019). Pay attention to features, transfer learn faster cnns. In *International Conference on Learning Representations*.