ABSTRACT
COLIEE is an annual competition in automatic computerized legal text processing. Automatic legal document processing is an ambitious goal, and the structure and semantics of the law are often far more complex than everyday language. In this article, we survey and report our methods and experimental results in using deep learning in legal document processing. The results show the difficulties as well as potentials in this family of approaches.

CCS CONCEPTS
• Computing methodologies → Neural networks; • Applied computing → Law.

KEYWORDS
Deep Learning, Legal Text Processing, JNLP Team

1 INTRODUCTION
COLIEE is an annual competition in automatic legal text processing. The competition uses two main types of data: case law and statute law. The tasks for automated models include: retrieval, entailment, and question answering. With deep learning models, JNLP team achieves competitive results in COLIEE-2021.

Task 1 is a case law retrieval problem. With a given case law, the model needs to extract the cases that support it. This is an important problem in practice. It is actually used in the attorney’s litigation as well as the court’s decision-making. Task 2 also uses caselaw data, though, the models need to find the paragraphs in the existing cases that entail the decision of a given case. Task 3, 4, 5 uses statute law data with challenges of retrieval, entailment, and question answering, respectively.

For the traditional deep learning approach, the amount of data provided by the organizer is difficult for constructing effective models. For that reason, we use pretrained models from problems that have much more data and then finetune them for the current task. In Tasks 1, 2, 3, and 4 we have used lexical score and semantic score to filter a correct candidate. In our experiments, the ratio is not 50:50 for lexical score and semantic score, we find out that the increase in the rate of lexical score lead to efficiency in ranking candidates.

Through problem analysis, we propose solutions using deep learning. We also conducted detailed experiments to explore and evaluate our approaches. Our proposals of deep learning methods for these tasks can be a useful reference for researchers and engineers in automated legal document processing.

2 RELATED WORKS
2.1 Case Law
In COLIEE 2018, most teams chose the lexical-based approaches. UBIRLED ranked the candidate cases based on tf-idf. They got approximately 25% of the candidate cases with the highest scores. UA and several teams also chose the same approach with UBIRLED. They compared lexical features between a given base case and corresponding candidate cases. JNLP team combined lexical matching and deep learning, which achieved state-of-the-art performance on Task 1 with the F1 score of 0.6545.

In COLIEE 2019, several teams applied machine learning including deep learning to both tasks. JNLP team achieved the best result of Task 1 in COLIEE 2019 [13] using an approach similar to theirs in COLIEE 2018. In Task 2, their deep learning approach achieved lower performance compared to their lexical approach. Team UA's combination of lexical similarity and BERT model achieved the best performance for Task 2 in COLIEE 2019 [8].

In COLIEE 2020, the Transformer model and its modified versions were widely used. TLIR and JNLP [7] teams used them to classify candidate cases with two labels (support/non-support) in Task 1. Team cyber encoded candidate cases and the base case in tf-idf space and used SVM to classify. They demonstrated the ability of the approach with the first rank in Task 1. In Task 2, JNLP [7] continually applied the same approach in Task 1 in the weakly-labeled dataset. It made them surpass team cyber and won Task 2.
2.2 Statute Law

The retrieval task (Task 3) is often considered as a ranking problem with similarity features. In COLIEE 2019, most of the teams used lexical methods for calculating the relevant scores. JNLP [12], DBSE [11] and IITP [1] chose tf-idf and BM25 to build their models. JNLP used tf-idf of noun phrase and verb phrase as keywords which show the meaning of statements in the cosine-similarity equation. DBSE applied BM25 and Word2vec to encode statements and articles. The document embedding was used to calculate and rank the similarity score. Team KIS [9] represented the article and query as vectors by generating a document embedding. Keywords were selected by tf-idf and assigned with high weights in the embedding process. In COLIEE 2020, with the popularity of Transformer based methods, the participants change their approaches. The task winner, LLNTU, only used BERT model to classify articles as relevant or not.

Regarding the entailment task (Task 4), approaches using deep learning have attracted more attention. In COLIEE 2019, KIS [10] used predicate-argument structure to evaluate similarity. IITP [1] and TR [4] applied BERT for this task. JNLP [5] classified each query to follow binary classification based on big data. In COLIEE 2020, BERT and multiple modified versions of BERT were used. JNLP [7] chose a pretrained BERT model on a large legal corpus to calculate the correctness of statements.

3. Method

3.1 Task 1 and Task 2. Case Law Processing

In 2021, COLIEE has changed a lot in the data structure of Task 1, and this year, the performance of participants' models drops badly from nearly 70% to 20% or worse. The particular reason for this circumstance is the huge searching space. For tackling this issue, we firstly used a lexical model to calculate the similarity between a given query and the whole case law corpus. Limiting the searching space to 100 candidate cases by picking the top 100 cases that have the highest BM25 scores for a given query.

The previous approaches focus on encoding the whole case law and calculating the similarity of vector representation for each query-candidate pair of cases. In our method, we handle the similarity between candidates on the paragraph level. "Supporting" and "Relevant" are classified as the paragraph level. "Supporting" and "Relevant" are classified as the paragraph level. In the query paragraph of the question, the paragraph is classified as "Supporting" if there is at least one paragraph in the candidate cases that has a higher BM25 score for a given query. If there is at least one paragraph in the candidate cases that has a higher BM25 score for a given query, the paragraph is classified as "Relevant."
Figure 2: Demonstration of mixing lexical and semantic score.

in the candidate paragraph number [31], we can see the semantic relationship between two paragraphs through this example. For discovering the relevant paragraph, we combine the lexical similarity and the semantic similarity score.

**Lexical Matching** For the exact purpose of capturing lexical information, we use Rank-BM25, a collection of algorithms for querying a set of documents and returning the ones most relevant to the query. For Task 1, after reducing the searching space to 100 queries, we separate the query case and also candidate cases into paragraphs. Assume that the query case has \( N \) paragraphs, and the candidate case has \( M \) paragraphs, we will calculate the lexical mapping score by Rank-BM25 for each query paragraph and every single paragraph in the candidate case. Then we obtain the matrix lexical score size \( N \times M \) for each query-candidate pair. We keep these matrixes and their union with supporting score which is introduced later.

**Supporting Matching** As we mentioned in the Figure 1, we want to extract the semantic relationship between query paragraph and candidate paragraph. To obtain this relationship score, we use pretrained model BERT, and this model is provided by huggingface. The same approach as lexical matching, we sequentially split a query and corresponding candidate cases into paragraphs, and obtain a matrix semantic score that has the same size as matrix lexical score. For more details about this supporting model, we do not use the original model provided by huggingface. Although BERT is currently one of the best, it is trained in the general domain. This actively demonstrates that BERT can not work well in a specialized domain such as law. For tackling this issue, we develop a silver dataset based on the COLIEE dataset for finetuning BERT and use this for predicting the semantic relationship between paragraphs.

**Union Score** To obtain the most relevant cases for a given query, we find a semantic as well as lexical relationship overlapped between the two paragraphs. For the exact purpose of developing union score we use the following formula:

\[
union\_score = \alpha \ast score\_supporting + (1 - \alpha) \ast score\_BM25
\] (1)

Figure 2 shows the architecture of the system.

For Task 2, we use the same approach as Task 1. For the final runs, we use the supporting model and lexical model for 2 runs, and in the last run, we use NFSP model [6]. Using our proposed approach, Task2 is basically a binary classification with the training data as a set of sentence pairs. As a result, we can obtain more gold training data to train the supporting model. For optimizing the performance of the supporting model, after finetuning on silver, we finetune one more time on gold data of Task 2.

3.2 Task 3. The Statute Law Retrieval Task

This task involves reading a legal bar exam question \( Q \), and extracting a subset of Japanese Civil Code Articles \( A_{i|1\leq i \leq n} \) which contains appropriate articles for answering the question, i.e. \( \text{Entails}(A_{i|1\leq i \leq n}, Q) \) or \( \text{Entails}(A_{i|1\leq i \leq n}, \text{not } Q) \).

As specified in [7], two main challenges of Task 3 are: (i) answering the questions which describe specific legal cases (note that the language used in statute law tends to be general), and (ii) addressing the long articles. The first challenge requires the model

---

1[https://pypi.org/project/rank-bm25/](https://pypi.org/project/rank-bm25/)

2[https://huggingface.co/models](https://huggingface.co/models)
to have deduction ability, which is a difficult task and requires further research. Regarding the second challenge, we addressed it by performing text chunking technique on the prepared training data and self-labeled technique while finetuning pretrained models.

**Training data preparation.** We follow the training data preparation method proposed in [7]. Naturally, the pair of a question and each of its annotated entailing articles is considered a positive training example, and the pair of a question and any other article is considered a negative training example. However, this approach results in very big negative-positive ratio in training data. To reduce this ratio, in this phase, the maximum number of negative training examples is limited to be 150, choosing based on the rank of tf-idf scores. Please refer to [7] for more details.

**Text chunking technique.** We use BERT and RoBERTa, which are two prevalent language models, as pretrained models. However, they cannot handle a very long article without truncating it, because of the 512-token limitation. Besides, in Task 3, we observe that, in most cases, only a few parts of the entailing article entails the corresponding question while other parts do not. We provide an example in Table 1, where the question is entailed by only one part of the article. To address the aforementioned limitation of BERT and RoBERTa, we propose to split each article into multiple chunks using a sliding window (as sliding windows mitigate the cases where the entailing part of the article is split). Regarding training data generation, we followed the method proposed in Nguyen et al. [7], except that the pair used for training is (question, chunk) instead of (question, article). The label of pair (question, chunk) is derived from the label of its corresponding pair (question, article).

![Table 1: An example that only one part of the entailing article entails the corresponding question.](image)

**Self-labeled technique.** Training data generated using text chunking contains noises. For example, assume that a long article A, which entails a question Q, is split into sub-articles A_1, A_2, ..., A_n, and only A_n entails Q, then the aforementioned training data generating method will label pairs (Q,A_1), (Q,A_2), ..., (Q,A_n) as positive training examples; however, only pair (Q,A_n) should be labeled positive, while other pairs, i.e. (Q,A_1), (Q,A_2), ..., (Q,A_{n-1}), should be labeled as negative examples. Inspired by the self-labeled techniques [14], we propose to deploy a simple self-labeled technique to help mitigate noisy training examples. Specifically, first, a pretrained model finetunes on the generated training data. Next, the finetuned model predicts labels for training examples. After that, we modify the labels based on rules, and the label-modified training data is used for the next finetuning phase. This self-labeled and finetuning process can be iterated multiple times. Regarding the label modifying rules, we only keep label modification if the label is converted from positive to negative. After label modification, the number of noisy examples tends to decrease, which allows the previous-phase finetuned model to learn from a more accurate data.

**Model ensembling.** The outcome of each model is the prediction based on the particular characteristics the models learned during training. Since each model has its advantages and disadvantages, model ensembling, an effective machine learning approach for incorporating models, appears to help produce better predictions. In our work, we ensemble models by deploying weighted aggregation on models’ predictions. Prior to the ensemble process, the outputs of multiple models were scaled by min-max normalization method, so that they were standardized in the range [0, 1]. It ensures that the standardized outputs of each model have a similar impact on the final prediction results. We divided the dataset into training, development, and test set. The ensemble method weights were constructed from the development set and applied to the test set.

### 3.3 Task 4. The Legal Textual Entailment Task

This task involves the identification of an entailment relationship between relevant articles A_{ij|s|sn} (which is derived from Task 3’s results) and a question Q. The models are required to determine whether the relevant articles entail "Q" or "not Q". Given a pair of legal bar exam question and article (Q, A_i), the models return a binary value for determining whether (A_i) entails (Q). To address this task, we modified the training data preparation step, then use the same model architecture in Task 3 (Section 3.2) for training.

**Data preparation.** Based on our observation, the challenge of this task is to extract the relevance between a question and articles for classification while the number of given articles is relatively small (usually 1 or 2 articles are given). We hypothesize that the model can extract information more effectively and consistently if there are more relevant articles given. Therefore, we adapt the data augmentation technique mentioned in Task 3 (which is based on tf-idf scores) to increase the number of relevant articles for each question. Besides, we also use the text chunking and self-labeled techniques introduced in Section 3.2 for dealing with long article challenge.

### 3.4 Task 5. Statute Law Question Answering

The goal in Task 5 of the models is to answer legal questions. In detail, with a given statement, the model needs to answer whether
Table 2: Examples of pretraining data.

| Sentence Pair                  | NFSP Label | NMS Label |
|-------------------------------|------------|-----------|
| Shall we go out? The weather is nice. | -          | 2         |
| いい 天気です。 出掛けしょう！ | -          | 1         |
| The weather is nice. Shall we go out? | 1      | 1         |
| The weather is nice. いい 天気ね。 | 0          | 0         |
| The weather is nice. Random Sentence. | 0      | 0         |
| いい 天気ね。 Shall we go out? | -          | 2         |

Table 3: Parameters and performances in pretraining the models on valid set.

| Model           | Max Len. | Batch Size | #Batches | Acc.  |
|-----------------|----------|------------|----------|-------|
| NFSP Base       | 512      | 16         | 24,000   | 94.4% |
| NFSP Distilled  | 512      | 32         | 34,000   | 92.2% |
| NMS Base        | 512      | 16         | 320,000  | 88.0% |
| NMS Distilled   | 512      | 32         | 496,000  | 87.7% |

Table 4: Supporting dataset.

| Data Source | #case | #paragraph | #sentence | #example |
|-------------|-------|------------|-----------|----------|
| Task 1      | 4415  | 172495     | 626540    | 378720   |
| Task 2      | 425   | -          | 913       | 18238    |

4 EXPERIMENTS

4.1 Task 1 and Task 2. Case Law Processing

Training data As we mentioned in Section 3, we create a supporting training dataset to train BERT model and the analysis of this training dataset is as Table 4. From 4415 cases in Task 1 raw dataset, we can extract more than 170K paragraphs. However, inside these paragraphs, some of them contain a lot of french text. This actively demonstrates that we need a filter step to extract clean English text. For the purpose of avoiding noise in the dataset, we use langdetect⁴ to filter and remove french text from training data. The clean paragraphs are obtained so far, we split every single paragraph into sentences (>625K in total), and from these sentences we apply some technique to generate supporting examples. The massive number of silver training examples for training BERT is over 350K examples.

Besides silver data, we utilize the data from Task 2 to extract more training dataset. As you can see in the Table 4, the number of gold examples we can extract is more than 18K.

For Task 1, we submit 3 runs as follow:
- Run1: Lexical score combine with semantic score with ratio α 7:3.
- Run2: Lexical score combine with semantic score with ratio α 3:7.
- Run3: Only supporting score.

For Task 2, we submit 3 runs as follow:
- Run1: Lexical score combine with semantic score with ratio α 7:3.
- Run2: Lexical score combine with semantic score with ratio α 7:3 and finetuning on gold training dataset.
- Run3: Lexical score combine with NFSP model’s score.

Our method on Task 1 has bad performance on the test set, the particular reason for this issue is the problem when we limit the searching space from 4000 to 100, maybe lexical matching works badly in this circumstance. For Task 2, as the numbers in Table 5, our method achieves 61% on F score, and we place 5TH in COLIEE 2021.

4.2 Task 3. The Statute Law Retrieval Task

We trained and evaluated our proposed models for this task with the previous year’s dataset. Macro-precisions, recalls, and F2 scores are reported. Note that the reported F2 scores are the F2 scores of class-wise precision means and class-wise recall means.

---

⁴https://pypi.org/project/langdetect/
| Run ID                          | F1 Score |
|--------------------------------|----------|
| BM25Supporting_Denoising       | 0.6116   |
| BM25Supporting_Denoising_Finetune | 0.6091   |
| NFSP_BM25                      | 0.5868   |

Table 5: Result on Task 2.

| Chunking info. | Return | Retrieved | P    | R    | F2  |
|----------------|--------|-----------|------|------|-----|
| no chunking    | 118    | 81        | 68.24| 72.52| 71.62|
| 110/20         | 190    | 108       | 61.20| 67.87| 66.42|
| 150/10         | 122    | 85        | 64.77| 66.67| 66.28|
| 150/20         | 129    | 93        | 68.09| 70.72| 70.18|
| 150/40         | 131    | 92        | 67.12| 71.17| 70.32|
| 150/50         | 132    | 94        | 69.74| 73.42| 72.66|
| 200/20         | 139    | 90        | 67.12| 72.97| 71.72|
| 300/30         | 110    | 74        | 65.39| 67.57| 67.12|

Table 6: (Task 3) Results of bert-base-japanese pretrained model finetuning with the text chunking technique. The values in the Chunking info. column denotes chunking settings with format <window_size>/<stride>.

We conducted multiple experiments with different settings to find the most appropriate settings when applying text chunking technique. In these experiments, we finetuned 3 epochs on bert-base-japanese pretrained model and report results in Table 6. We use <window_size>/<stride> to denote the sliding window parameters. The results indicate that <150>/<50> seems to be the most appropriate setting for the task. We use this setting to conduct experiments relating to self-training technique. Specifically, first, we finetuned bert-base-japanese pretrained model with e1 epochs, then performed self-labeling process and continued to finetune with e2 epochs. We use e1/e2 to denote this settings. The results in Table 7 suggests that e1 = 2 seems to be the "just right" parameter for the first finetuning process (when e1 = 3, the finetuned model seems to start overfitting); and as e2 increases, F2 tends to increase. The results demonstrate the positive impact of our proposed methods: F2 increased from 71.62 to 72.66 when applying the text chunking technique, and this number is 72.91 with the self-labeled technique.

We also did experiments with bert-base-japanese-whole-word-masking and xlm-roberta-base pretrained models, but we do not report in this paper because of paper length limitation. We performed model ensembling on the outputs of those models.

For the contest submissions, we submitted 3 runs based on the 3 proposed approaches. The model ensembling method returned the highest result among the three. We are the runner-up of this task. The results of final runs of all participants are in Table 8.

### 4.3 Task 4. The Legal Textual Entailment Task

Similar to Task 3, we also trained and evaluated our proposed models with the dataset in the previous year with the questions having id R-01-" as the development set. Because of the relatively small training data, we run 5 times with each setting and report the mean and standard deviation values. For this task, most of the hyperparameters follows settings in Nguyen et al. [7] where batch_size = 16 and learning_rate = 1e-5.

**Pretrained model and Data augmentation.** Firstly, we conducted the experiments to find the most suitable pretrained model for this task, and the most suitable setting for the tf-idf-based data augmentation method (Table 9). Based on the experimental results, we found that the bert-base-japanese-whole-word-masking pretrained model is more suitable for this task than others. Besides, the tf-idf-based augmentation data method also help increase the model performance.

**Performance stability.** In addition, we found that the performance of the model with a small epoch is fairly unstable. Therefore, we experimented with a bigger number of training epochs (Table 10). The experimental results demonstrate that the runs with higher epochs tend to achieve more stable accuracies, but the model

| Run ID             | sid F2 | Prec | Recall |
|--------------------|--------|------|--------|
| OvGU_run1          | E/J    | 0.7302| 0.6749| 0.7778 |
| JNLP.CrossLMultiThreshold | E/J | 0.7227| 0.6000| 0.8025 |
| BM25.UA            | E/J    | 0.7092| 0.7531| 0.7037 |
| JNLP.CrossLBertQP  | E/J    | 0.7090| 0.6241| 0.7116 |
| R1.LLNTU           | E/J    | 0.7047| 0.6656| 0.7438 |
| R2.LLNTU           | E/J    | 0.7039| 0.6770| 0.7035 |
| R1.LLNTU           | E/J    | 0.6875| 0.6368| 0.7315 |
| JNLP.CrossLBertJPC15030C15050 | E/J | 0.6838| 0.5535| 0.7778 |
| OvGU_run2          | E/J    | 0.6717| 0.4857| 0.8025 |
| TFDFAU             | E/J    | 0.6571| 0.6790| 0.6543 |
| LM.UA              | E/J    | 0.5460| 0.5679| 0.5432 |
| TR_HB              | E/J    | 0.5226| 0.3333| 0.6173 |
| HUKB-3             | J      | 0.5224| 0.2901| 0.6975 |
| HUKB-1             | J      | 0.4732| 0.2397| 0.6543 |
| TR_AV1             | E/J    | 0.3599| 0.2622| 0.5123 |
| TR_AV2             | E/J    | 0.3369| 0.1490| 0.5556 |
| HUKB-2             | J      | 0.3258| 0.3272| 0.3272 |
| OvGU_run3          | E/J    | 0.3016| 0.1570| 0.7006 |

Table 8: (Task 3) Result of final runs on the test set, the underlined lines refer to our models.
Table 9: (Task 4) Model accuracies with different tf-idf augmentation setting. The name BertJp, BertJp2 indicates that we used a pre-trained bert-base-japanese and bert-base-japanese-whole-word-masking, respectively. The name column follows the format tf-idf<number> where <number> denotes the number of augmented articles appended for each question.

| Model       | Origin            | tf-idf1 | tf-idf2 | tf-idf5 | tf-idf20 |
|-------------|-------------------|---------|---------|---------|---------|
| BertJp      | 55.9±3.7          | -       | -       | -       | -       |
| BertJp2     | 60.4±4.1          | 61.6±5.0| 62.7±5.7| 61.3±4.4| 58.4±2.9|

Table 10: (Task 4) Accuracies of BertJp2 with different training epochs.

| # epochs | tf-idf1 | tf-idf2 | tf-idf5 |
|----------|---------|---------|---------|
| 3        | 61.6±5.0| 62.7±5.7| 61.3±4.4|
| 10       | 61.8±3.0| 62.9±2.2| 64.7±2.5|
| 20       | -       | 61.6±1.8| 64.0±1.3|

Table 11: (Task 4) Accuracies of BertJp2 using the self-labeled technique with different settings on chunking data where window_size = 150, stride = 50. The values in the Setting column denotes \(<e_1/>/<e_2>/\).

| Setting | tf-idf2 | tf-idf5 |
|---------|---------|---------|
| 1/10    | 62.9±1.7| 63.1±1.9|
| 2/10    | 64.3±0.5| 64.3±0.5|
| 3/10    | 62.2±1.7| 64.1±1.0|

can be overfitted if we increase the number of epochs too much. Besides, the augmentation data also helps the model performance to be more stable.

Long article challenge. Finally, to address the long article challenge, we conducted the experiments using the text chunking and the self-labeled techniques described in Task 3 (Table 11). Although the accuracy of models in this setting did not increase, the variant ranges are smaller. It may be because the text chunking and the self-labeled techniques help eliminate noises in training data.

The results on blind test set are shown in the Table 12 with the id "JNLP.Enss5C15050" refers to the model BertJp2 using augmentation data tf-idf5, "JNLP.Enss5C15050SilverE2E10" refers to the model BertJp2 using augmentation data tf-idf5, and \(<e_1/>/<e_2>/\) is 2/10; "JNLP.EnssBest" refers to the ensemble of both models.

Table 12: (Task 4) Results final runs on the test set. The underlined lines refer to our submissions.

| Model       | Accuracy  |
|-------------|-----------|
| NFSP Base   | 71.0%     |
| NFSP Distilled | 51.1%   |
| NMSP Base   | 79.5%     |
| NMSP Distilled | 48.9%   |
| XLM-RoBERTa | 51.1%     |
| BERT Multilingual | 64.1% |

Table 13: (Task 5) Performance of models on validation set.

| Model       | Accuracy  |
|-------------|-----------|
| Baseline    | 0.5369    |
| JNLP.NFSP   | 0.6049    |
| UA_parser   | 0.5679    |
| JNLP.NMSP   | 0.5556    |
| UA_d1       | 0.5556    |
| TR_Ensemble | 0.5432    |
| KIS_2       | 0.5062    |
| KIS_3       | 0.5062    |
| UA_elmo     | 0.4938    |
| JNLP.BERT_Multilingual | 0.4691 |
| KIS_1       | 0.4321    |
| TR_GPT3Ada  | 0.4321    |
| TR_GPT3Davinci | 0.4321 |

Table 14: (Task 5) Result of final runs on the test set, the underlined lines refer to our models.

Therefore, we choose NFSP Base, NMSP Base and original BERT Multilingual as candidates for the final run.

4.4 Task 5. Statute Law Question Answering

We compare our proposed models together and with other cross-lingual and multilingual baselines such as XLM-RoBERTa [2] and original BERT Multilingual [3]. In the 7000 augmented sentences, we divide the train set and validation set with the ratio of 9:1.

Our experiments show that NFSP Base and NMSP Base achieve the best performance and have stable loss decrease. NFSP Distilled, NMSP Distilled and XLM-RoBERTa fail to learn from the data and their performance equivalent to that of random sampling. BERT Multilingual is in the middle of the ranked list of models in Table 13.

5 CONCLUSIONS

This paper presents JNLP Team’s approaches using deep learning to the legal text processing tasks in the COLIEE-2021 competition.
Due to the limited amount of data and the difficulty of the tasks, we used pretraining methods to solve the problems. The experimental results and the performance of the models on the blind test show the reasonability and robustness of the proposed methods.

ACKNOWLEDGMENTS
This work was supported by JSPS KAKENHI Grant Numbers JP17H06103 and JP20K20406.

REFERENCES
[1] B. Gain, D. Bandyopadhyay, T. Saiikh, and A. Ekbal. 2019. Iitp@coliee 2019: Legal information retrieval using bm25 and bert.
[2] Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Unsupervised cross-lingual representation learning at scale. arXiv preprint arXiv:1911.02116 (2019).
[3] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805 (2018).
[4] J. Hudzina, T. Vacek, K. Madan, C. Tonya, and F. Schilder. 2019. Statistical entailment using similarity features and decomposable attention models.
[5] HT Nguyen, V. Tran, and LM Nguyen. 2019. A deep learning approach for statute law entailment task in COLIEE-2019. Proceedings of the 4th Competition on Legal Information Extraction/Entailment. COLIEE (2019).
[6] Ha-Thanh Nguyen, Vu Tran, Phuong Minh Nguyen, Quan Minh Bui, Chau Minh Nguyen, Binh Tran Dang, Hai-Yen Thi Vuong, Ken Satoh, and Minh Le Nguyen. 2021. ParaLaw Nets - Cross-lingual Sentence-level Pretraining for Legal Text Processing.
[7] Ha-Thanh Nguyen, Hai-Yen Thi Vuong, Phuong Minh Nguyen, Binh Tran Dang, Quan Minh Bui, Sinh Trong Vu, Chau Minh Nguyen, Vu Tran, Ken Satoh, and Minh Le Nguyen. 2020. JNLP Team: Deep Learning for Legal Processing in COLIEE 2020. arXiv preprint arXiv:2011.08071 (2020).
[8] Juliano Rabelo, Mi-Young Kim, and Randy Goebel. 2019. Combining Similarity and Transformer Methods for Case Law Entailment. In Proceedings of the Seventeenth International Conference on Artificial Intelligence and Law (ICAIL ’19). 290–296. https://doi.org/10.1145/3322640.3326741
[9] R. Hayashi and Y. Kano. 2019. Searching relevant articles for legal bar exam by doc2vec and tf-idf.
[10] R. Hoshino, N. Kiyota, and Y. Kano. 2019. Question answering system for legal bar examination using predicate argument structures focusing on exceptions.
[11] S. Wehnert, S.A. Hoque, W. Fenske, and G. Saake. 2019. Threshold-based retrieval and textual entailment detection on legal bar exam questions.
[12] T.B. Dang, T. Nguyen, and L.M. Nguyen. 2019. An approach to statute law retrieval task in coliee-2019.
[13] Vu Tran, Minh Le Nguyen, and Ken Satoh. 2019. Building legal case retrieval systems with lexical matching and summarization using a pre-trained phrase scoring model. In Proceedings of the Seventeenth International Conference on Artificial Intelligence and Law. 275–282.
[14] Isaac Triguero, Salvador García, and Francisco Herrera. 2015. Self-labeled techniques for semi-supervised learning: taxonomy, software and empirical study. Knowledge and Information Systems 42, 2 (2015), 245–284.