Wehnert, Sabine; Dureja, Shipra; Kutty, Libin; Sudhi, Viju; De Luca, Ernesto William

Applying BERT Embeddings to Predict Legal Textual Entailment

The Review of Socionetwork Strategies

Provided in Cooperation with:
Springer Nature

Suggested Citation: Wehnert, Sabine; Dureja, Shipra; Kutty, Libin; Sudhi, Viju; De Luca, Ernesto William (2022) : Applying BERT Embeddings to Predict Legal Textual Entailment, The Review of Socionetwork Strategies, ISSN 1867-3236, Springer Nature Singapore, Singapore, Vol. 16, Iss. 1, pp. 197-219,
https://doi.org/10.1007/s12626-022-00101-3

The Review of Socionetwork Strategies (2022) 16:197–219
https://doi.org/10.1007/s12626-022-00101-3

ARTICLE

Applying BERT Embeddings to Predict Legal Textual Entailment

Sabine Wehnert[1,2] · Shipra Dureja[2] · Libin Kutty[2] · Viju Sudhi[2] · Ernesto William De Luca[1,2]

Received: 19 September 2021 / Accepted: 7 January 2022 / Published online: 19 February 2022

Abstract

Textual entailment classification is one of the hardest tasks for the Natural Language Processing community. In particular, working on entailment with legal statutes comes with an increased difficulty, for example in terms of different abstraction levels, terminology and required domain knowledge to solve this task. In course of the COLIEE competition, we develop three approaches to classify entailment. The first approach combines Sentence-BERT embeddings with a graph neural network, while the second approach uses the domain-specific model LEGAL-BERT, further trained on the competition's retrieval task and fine-tuned for entailment classification. The third approach involves embedding syntactic parse trees with the KERMIT encoder and using them with a BERT model. In this work, we discuss the potential of the latter technique and why of all our submissions, the LEGAL-BERT runs may have outperformed the graph-based approach.

Keywords Contextual embeddings · Graph embeddings · Transformers · Textual entailment

1 Introduction

In this work, we develop three approaches for legal textual entailment classification on the English version of the Japanese Civil Code. This research is part of task 4 of the Competition on Legal Information Extraction/Entailment (COLIEE). The task consists of two texts which are compared to decide on a binary entailment relationship. In this case we have a query and one or multiple associated articles from the English version of the Japanese Civil Code.

✉ Sabine Wehnert
sabine.wehnert@gei.de

[1] Leibniz Institute for Educational Media | Georg Eckert Institute, Brunswick, Germany

[2] Otto von Guericke University Magdeburg, Magdeburg, Germany

In general, textual entailment classification requires capabilities which are normally attributed to humans who can acquire a deep knowledge of the legal domain to understand and interpret legal texts to reason about their relationship and lawfulness. Such reasoning capabilities are yet to be developed on a machine, for example as a decision support in specific legal cases. International activities make it hard—even for legal professionals—to oversee all legislations which may be relevant for a specific case and to determine compliance with the law via entailment. Therefore, we perform research on this topic towards the goal of pushing the limits of current Legal AI approaches. With the advent of deep learning, there are many models which are tested on natural language inference tasks, and the same development exists in the COLIEE competition. Although their decision making is hard to understand for a human, deep learning approaches have consistently achieved good results in the past years on this task. They are often outperforming more explainable methods and because they are not trusted in the legal domain, the research and detailed analysis of their strengths and weaknesses is important to understand future research directions.

In particular, the BERT model (Bidirectional Encoder Representations from Transformers) has achieved good scores in the past COLIEE editions. Aside from ongoing state-of-the-art performance of BERT variants on many tasks in natural language processing, BERT offers contextual word embeddings which are an advancement of distributional semantic approaches. Previous approaches often failed to correctly encode the contextual meaning of a word. Therefore, using BERT in the COLIEE competition to overcome challenges, such as term mismatch and different abstraction levels of the two documents to be compared can be helpful. Hence, we rely in our three approaches on some variant of the BERT model. Nowadays, it is almost a standard procedure to choose a domain-specific pre-trained BERT model and then to fine-tune it on a downstream task.

Using only a BERT model though will not solve this task and has been done before. One part of our work is motivated by the recent advancements in Graph Neural Networks (GNNs), which can also be combined with the BERT model. Since the relationship between the query and article is a relevance relationship, the data can be transformed into a graph format to encode structural information about the relationships between nodes and their individual features. Another major challenge in the COLIEE competition is the size of the dataset, with 806 instances to train a model. A commonly mentioned drawback of deep neural networks is the data size which is required to learn meaningful feature representations. In our work, we study also data augmentation and enrichment to work with our graph-based and BERT-based deep learning approaches despite the small dataset size.

In addition, the required language modeling capabilities to solve the entailment task may go beyond the contextual representations we obtain from language models such as BERT, which are based on distributional semantics. Injecting linguistic knowledge in such a model may be a worthwhile consideration, given that some instances contain paraphrases of verbs or nouns which may be easier identified if the parse trees provide syntactic features, potentially easing the alignment of query and article. Therefore, we also employ KERMIT (Kernel-inspired Encoder with Recursive Mechanism for Interpretable Trees) [30] which can make use of symbolic

syntactic parse trees as additional features to the contextual embeddings of a BERT model.

Based on our experiments, we offer insights in our result analysis, discuss some challenges we faced in this competition and draw conclusions for future research.

With this paper, we make the following contributions:

- We employ an ensemble of Graph Neural Networks together with features from Sentence-BERT and metadata of the Civil Code for the task.
- We perform pre-training on the statute law retrieval task and data decomposition to improve the learning of a domain-specific model called LEGAL-BERT.
- We test the KERMIT+BERT architecture for encoding syntactic parse trees combined with a BERT model to inject further linguistic knowledge.

After the introduction, the remainder of this work is structured as follows: In Sect. 2, we collect approaches of contextual embeddings from language models and graph embeddings for entailment classification. In Sect. 3, we describe the concepts of our approaches for the ensemble of Graph Neural Networks (GNNs) in our submitted run 1 and LEGAL-BERT in run 2 and 3 of the competition, as well as newer experiments with the KERMIT+BERT architecture. Section 4 contains our evaluation setting, results and their analysis within a discussion. We conclude this work and indicate future research in Sect. 5.

2 Related Work

2.1 Contextual Embeddings from Language Models

With the growth in applications for Natural Language Processing (NLP), various fields of software technology such as machine translation, text recognition and text generation have seen a large development in the area of deep learning models adapting to these tasks [8]. Substantial progress in the area of learning embeddings for dense representations of the textual data has been made. Some of them are CoVe [16] (Contextual Word Vectors), ELMo [19] (Embeddings from Language Model), Cross-View Training [3] (CVT), ULMFiT [10] (Universal Language Model Fine-tuning for Text Classification), GPT [22] (Generative Pre-training Transformer), BERT [4], ALBERT [14] (A Lite BERT) and RoBERTa [15] (Robustly optimized BERT approach). These dense representations are usually learned by training on auxiliary tasks, such as masked language modeling (MLM), next sentence prediction (NSP), machine translation, and transcription. Contextual word embeddings—once learned—can be further fine-tuned for downstream tasks, such as classification, with relatively less effort. The language model BERT [4] and its variants [14, 15] have emerged as the most convenient choice for a model concerning these downstream tasks since they condition a word's embedding on the surrounding context. This makes them and similar approaches perform significantly better than other models which learn static embeddings as a dense representation of the textual data.

One noticeable characteristic of BERT is that it performs better on domain-specific tasks when pre-trained with data of that specific domain. Various examples include *BioBERT-cased*, *PubMedBERT-uncased* which both perform better for biomedical data than the original BERT model as discussed by Gu et al. [6]. Similarly, LEGAL-BERT and its variants on legal sub-domains can perform better than the standard BERT on domain-specific tasks as summarized by Chalkidis et al. [2]. For LEGAL-BERT pre-training is carried out on a collection of several fields of English legal text like contracts, court cases, and legislation. The *legal-bert-base-uncased* model [2] is similar to the standard English *bert-base-uncased* model [4] in its neural network architecture. It has 12 layers, 768 hidden units, 12 attention heads, and 110M parameters. The pre-training is carried out with 1 million training steps on batch sizes of 256 with a maximum sequence length of 512 starting with a learning rate of 1e-4. It also has a similar training procedure to *bert-base-uncased*.

Supplying further syntactic knowledge to a BERT model is a recent field of study and has given promising results on other tasks [12, 30]. Although there are indications that the BERT model itself already encodes syntactic information implicitly, the KERMIT*viz* architecture enables visualizing which part of the sentence is used during inference, and using the KERMIT encoder in addition to a transformer has outperformed the standalone models BERT and XLNet on several tasks [30]. Therefore, KERMIT may have the potential to enhance both model performance and interpretability of the predictions on the COLIEE dataset.

In the previous year of the COLIEE competition, several teams used pre-trained BERT-based models with variations to address this entailment task [20]. The team *CU* submitted two such models. For the first run, they selected the *bert-multilingual-cased* model for sequence classification and then fine-tuned it on training data provided by COLIEE organizers. The other model was additionally trained on articles obtained from a term frequency - inverse document frequency (TF-IDF) model for the retrieval task 3. A closely related approach is the submission of Team *CYBER*. They use a pre-trained RoBERTa instead of *bert-base-uncased* and fine-tune it on the SNLI dataset followed by the COLIEE dataset to entail relevant articles. The team JNLP [18] also focused on the BERT-based approach to submit three runs, one of them gained the winning accuracy in COLIEE 2020 for task 4. We use their winning run as a motivation for our runs 2 and 3. They use domain-specific pre-training of BERT with American case law data with a corpus of 8.2M sentences. This model was then fine-tuned for addressing the lawfulness classification problem using additional augmented data of the English version of the Japanese Civil Code and COLIEE training data. To summarize this part, the choice of a suitable pre-trained model with subsequent fine-tuning or additional encoding of syntactic features can have a big impact on the success of a BERT-based approach for entailment classification.

2.2 Graph Embeddings

Graph Neural Networks (GNNs) are popular for data which can be represented by relations in a graph format. GNNs are used in many fields, such as computer vision, natural

language processing and combinatorial optimization. In the field of NLP, GNNs solve tasks such as text classification, question answering and entity retrieval. Yao et al. [27] have used a Graph Convolutional Network for text classification by forming a graph from word co-occurrences and document-word relations. They achieve scores which beat state-of-the-art methods on standard text classification benchmark datasets. De Cao et al. [1] use GNNs for question answering with named entities as nodes and edges as relations between the nodes. Xu and Yang [26] build a coreference resolver by encoding text with a BERT model and forwarding it to a fully connected layer, which is later concatenated with another feature representation obtained from a GNN. In particular, they receive the other GNN-based representation by combining the BERT encoding also as a feature with a syntactic dependency graph. This is then the input to a relational Graph Convolutional Network. Since we also combine a GNN with a BERT model, this approach is the most related to our first run. To the best of our knowledge, we are the first team using a GNN approach in the COLIEE competition task 4.

In particular, we consider a GNN architecture which is called Message Passing Neural Network (MPNN). Those networks operate with differentiable functions in two phases: the former being a message passing phase with defined message functions at each timestep, as well as a node update function, whereas the latter is a readout phase, having a readout function that computes the feature vector of the whole graph [5]. The advantages of this technique are a possibly higher adaptation and generalization capability of those networks due to their use of aggregated local neighborhood information [17]. In our case, we will not make use of the readout phase because instead of the whole graph's embedding, we feed the hidden states of the updated nodes individually into a classification model, which we describe in Sect. 3.1.

BERT-based approaches prove to be an effective solution for the past COLIEE edition. Moreover, domain-specific models may achieve better results than their standard variants. Graph Neural Networks have not been widely explored in the COLIEE competition. Hence, we focused on the use of LEGAL-BERT and GNNs for the COLIEE 2021 challenge on the statute law entailment task. Furthermore, we perform preliminary test of a KERMIT+BERT architecture on the COLIEE 2021 dataset.

3 Statute Entailment Task

For the competition, we develop two different approaches with three different runs, as described in Table 1. The first technique is an ensemble of GNNs, while the second and the third runs use LEGAL-BERT with different training approaches. In addition, we test the KERMIT architecture in a BERT model on the same dataset.

3.1 Ensemble of Graph Networks

Our graph consists of a set of nodes and edges, where each node represents either a query or an article. Edges are the connections between nodes. In the context of classification tasks, such a graph is often encoded by a neural network. This results in a graph embedding, which is then used as a feature, for example of a linear classification

Table 1 Methods for each run for task 4

Run	Method
OvGU_run1	Ensemble of Graph Neural Networks
OvGU_run2	LEGAL-BERT
OvGU_run3	LEGAL-BERT

layer. Standard graph embedding approaches focus on the structure of the graph only. However, encoding external knowledge into such a graph as node features and using them while creating a graph embedding can be helpful, especially in cases of a rather small graph. This is particularly interesting for the statute entailment task with limited training data, and where the contextual meaning of queries and articles can be found in their content, but also within the relation between them. Graph approaches can model abstract relations which cannot be easily characterized, such as the entailment relationship. So we decide to use graphs in connection to contextual word embeddings to encode relationships between a query and positively or negatively entailed articles.

In our implementation, we form a bipartite directed graph between the article and query nodes and try to learn the relation between them. Bipartite graphs can be divided into two sub-graphs with each sub-graph having no connections within itself but connections with the nodes of the other sub-graph. We choose this type of graph because we cannot directly model a relationship between the queries, however, the training data indicates a positive or negative entailment relationship between a query and its relevant articles. Since graphs can encode multiple relations to one node, we assume this is a good approach with the COLIEE dataset, where queries have multiple relevant articles and learning semantic relations within each of these query-article pairs can be beneficial for entailment classification.

Our Graph Neural Networks employ a message passing technique M and two different node update functions V_1 and V_2. V_1 represents the query node update function and V_2 represents the article node update function. Regarding the concept of message passing, the so-called message from one node is passed to another node, embedding the neighborhood information of the node with an aggregation function. Here, we use the average aggregation function [11] for message passing. Equation 1 shows our message passing function M where W represents parameter weights, ew stands for edge weight and x is the article/query node feature/embedding. Equation 2 shows the query node update function V_1 where W stands for parameter weights, x_q denotes the query node feature/embedding and m_a represents the article aggregation value. Equation 3 shows the article node update function V_2 where x_a represents article feature and m_q represents the query aggregation.

$$M(ew, x) = ew * W * x \tag{1}$$

$$V_1(x_q, m_a) = concatenate(W * x_q, m_a) \tag{2}$$

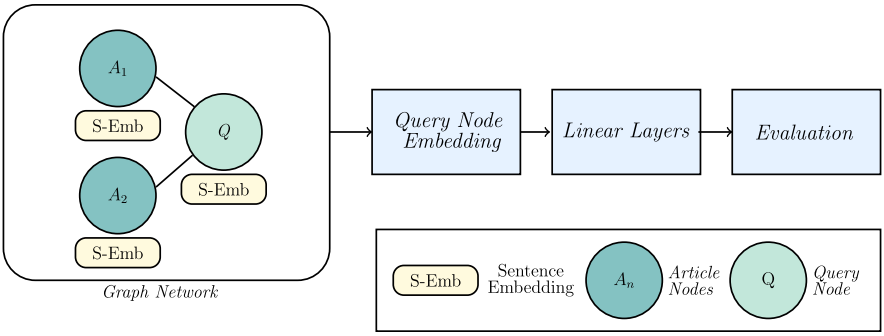

Fig. 1 Workflow for the Ensemble of Graph Methods

$$V_2(x_a, m_q) = x_a + m_q \tag{3}$$

In the implementation by He et al. [7], they have used a large scale bipartite graph to get an efficient representation with intra- and inter-message aggregation. We use the inter-message aggregation method to just pass article information to the queries and not to other article nodes.

We propose two graph structures differing in their message passing modality to learn the entailment relationship, as discussed below:

1. **Unidirectional message passing**: In this structure, we pass a message from the articles to their corresponding query and learn the embeddings. Equation 4 shows how we calculate embeddings for a query node which is in line with the work by Morris et al. [17], except for the difference that we concatenate the neighbour information instead of adding them, which has been done in the original implementation. The connection between the nodes is indicated in Fig. 1, however in this message passing variant the article node has a uni-directional connection to a query node and not the other way around. Let

$$x_q = \sigma(V_1(x_q, \frac{1}{|N|}\sum_{j=1}^{N} M(e_{q,a_j}, x_{a_j})), \tag{4}$$

   where x_q represents query features as the node embedding, x_a represents relevant article features, N is the number of articles associated with a query by a positive or negative entailment relationship, and $e_{q,a}$ represents the edge weight between query and article.

   For σ, we selected the rectified linear unit as an activation function.
2. **Bidirectional message passing**: In this structure, we make use of queries in the training dataset which hold an entailment relationship with the articles to update the article embeddings (both positive and negative) motivated by the approach of Wehnert et al. [25]. Initially, we update the articles with the related queries and then update the queries with the related articles. We suppose this could further enrich the article embeddings, since the entailed queries can add more unique

Table 2 Example of metadata for Article 567 of the Civil Code

Training data
Article 567: (1) If the seller delivers the subject matter .. ground of the loss or damage.
In such a case, the buyer may not refuse to pay the price.
(2) The preceding paragraph also applies if the seller tenders .. tender of the performance
due to any grounds not attributable to either party.
Metadata
Part: III Claims *Chapter:* II Contracts *Section:* 3 Sale
Sub-section: (Transfer of Risk for Loss of Subject Matter)

information about the articles. This is depicted in Fig. 1, where the article node is connected to a query node and also, the other way around. Equation 4 holds for this graph structure as well, with the modification of the article embedding x_a as below,

$$x_a = V_2(x_a, \frac{1}{|N_q|} \sum_{j=1}^{N_q} \beta * M(e_{a,q_j}, x_{q_j})), \tag{5}$$

where N_q refers to the number of queries associated with the article and β is a hyper-parameter which indicates the degree of influence of the query to the article. We have considered 0.1 as the β value, so that query should not influence the articles much to avoid an overflow of information from one query to another. Figure 1 shows the workflow[1] of how the entailment is done with the Graph Neural Network. Each article and query node in the graph is represented by features. We use extracted metadata from the section titles for each article to enrich its content before further processing with the language model. Table 2 gives an example of the metadata we used. With Sentence-BERT[2] [23], we generate sentence embeddings from the content of the queries and of the enriched articles and consider them as a feature for the corresponding node in the graph. We used the pre-trained *paraphrase-distilroberta-base-v1* model to create the sentence embeddings because it is claimed to work well on natural language inference tasks and was trained on millions of paraphrase pairs[3].

The starting point for the node embeddings are the sentence embeddings. In the unidirectional message passing scenario, we update those embeddings with the above function 4 for the query nodes only, based on the implementation by Morris et al. [17]. The resulting query node embedding also encodes information considering relevant articles as direct neighbor nodes. In the bidirectional case, we use

[1] Note that we use either a unidirectional or a bidirectional graph structure for our experiments, however, we have depicted this with a single line in Fig. 1 for brevity.

[2] https://github.com/UKPLab/sentence-transformers.

[3] https://www.sbert.net/docs/pre-trained_models.html#paraphrase-identification.

message passing to update both the query and article node sentence embedding features with functions 4 and 5, respectively. We then use the node embeddings for the downstream task of query node classification for entailment.

For our run 1, we submitted the unidirectional message passing model in an ensemble setup as follows: We train two different models with different parts of the dataset and ensemble their results to get the final entailment relationship. We train one model with the training dataset except for instances starting with the pair ID "R01-*" and another model with the full training dataset. In the time of inference, we take the average of the softmax values from the two models and hence determine the final prediction.

To summarize, we used a novel approach by employing sentence embeddings and node embeddings together to solve the entailment task. Also, we enriched the data with structural information of the Civil Code. In the following, we present the approach of using LEGAL-BERT for the two other runs.

3.2 LEGAL-BERT

For our runs 2 and 3, we consider a pre-trained model called *legal-bert-base-uncased*[4] [2] as our default choice for this task. More details about how that pre-trained model is obtained have been shared in Sect. 2.1. We also participated in task 3 of COLIEE, the statute law retrieval task, and fine-tuned the aforementioned BERT model for that purpose. In that work [25], LEGAL-BERT outperformed the regular BERT model (*bert-base-uncased*) and another domain-specific variant called *legal-RoBERTa*, so we did not employ the other models on task 4 anymore. The query-article pairs in the training data are the same for both tasks, but for task 4, we have additional entailment labels which are not required in task 3. However, since we performed task 3 also as a classification task, we used a different set of labels that described the relevance of an article to a query rather than the entailment labels. For task 3, except for the classification head, the encoder part of *legal-bert-base-uncased* is trained on the query-article pairs. In short, we transformed the originally imbalanced dataset in task 3 by decomposing all relevant articles into separate instances, in addition to keeping the original instance with potentially multiple relevant articles also in the training data. This increases the count of examples which contain relevant articles. Although this method does not necessarily result in correct entailment relationships among the decomposed instances, it helps in training the model in presence of only few samples. Then, per query, we add the top 50 most similar, but not relevant articles to reduce the overall amount of irrelevant articles and thus include not obvious cases in the training phase for each query. We leveraged this trained encoder for task 4 and re-initialize the classification head with new trainable parameters.

In detail, the following steps were done to preprocess the training data:

[4] https://huggingface.co/nlpaueb/legal-bert-base-uncased.

1. **Data decomposition:** The training data consists of query-article(s) pairs, where each query has one or more relevant articles. To extend the dataset for training, we split the training instances such that every instance has just one relevant article to which the given query is entailed. This increases the number of training instances and aids the model during training.
2. **Data augmentation with non-relevant articles:** Further, for each instance in the new decomposed training dataset, we checked the cosine similarity of the relevant article in that instance with all the articles in the Civil Code, except for the ones which were already marked relevant by another instance of the same query. We used TF-IDF vectorization to compute the similarity and picked the top 50 articles. These are the potential non-relevant articles for the query but closely related to its relevant articles. This extension is motivated by the implementation by Nguyen et al. [18], however, they considered query-article similarity instead of the article-article similarity that we propose. We proposed the article-article similarity because we believe that articles are more closely associated with each other than they are with the queries, such that model can benefit from non-relevant examples that can have a similar terminology to the relevant ones.
3. **Augmenting the dataset:** On top of this decomposed and augmented dataset, we also consider the training instances in the training dataset in their original form. This also introduces duplicate instances but ensures that the original query-article(s) pair could still influence the model.

Note that both steps 2 and 3 for augmenting data are performed only for task 3, the statute retrieval task, from which we adopt the encoder model, and for task 4 we only perform that data decomposition step. Anyway, we describe all three steps as we adopted the task 3 encoder for task 4.

An example of our query-article instance in the training dataset is given in Table 3 for the query of pair id *H27-22-1*:

Query Q: "In the case of a contract for sale of a specified thing, if the performance of the delivery has become impossible due to reason attributable to the seller, the effect of the contract of sale shall be lost by operation of law, then the buyer shall be relieved of liability for payment of the purchase money."

We adopt two different training techniques to generate the run 2 and run 3 of our submission as depicted in Fig. 2. The reason for this approach is that in previous experiments, we observed differences in model performance depending on the training and test split. This is due to a shift in the distribution of the problem categories, as we compile the statistics for the years 2018-2020 from the official summary papers [20, 21, 28] in Table 4. Some models may fit to a particular year's problem type distribution better than in the year after, however the performance over the years is not really comparable, as the dataset is extended in every new COLIEE edition. The official problem category assignments are not released, however we find descriptions of each category in the work by Hoshino et al. [9], which helps us to analyze our contributions in Sect. 4 based on those descriptions.

We also experiment with different hyper-parameters, the results of this are discussed in the evaluation section. For both runs, we create a training and validation split (all queries starting with the ID "R01-*" for the validation set). For run

Table 3 Data Decomposition to create additional instances using multiple relevant articles for each query

Queries	Articles	Label
Before preprocessing		
Query Q	**Article 415** (1) If an obligor fails to perform ... (2) ... obligor's failure to perform the obligation **Article 542** (1) In the following cases, the obligee may ... (2) . perform the part of the obligation **Article 543** If non-performance of an obligation is ... contract under the preceding two Articles.	Y
After preprocessing		
Query Q	**Article 415** (1) If an obligor fails to perform ... (2) ... obligor's failure to perform the obligation	Y
Query Q	**Article 542** (1) In the following cases, the obligee may ... (2) . perform the part of the obligation	Y
Query Q	**Article 543** If non-performance of an obligation is ... contract under the preceding two Articles.	Y

Table 4 Top 5 Ranking problem categories in the 2018–2020 test data with the number of instances

Year	Rank 1	Rank 2	Rank 3	Rank 4	Rank 5	Total
2018	Conditions	Person role	Person relationship	Morpheme	Anaphora	
	31	27	26	25	20	69
2019	Conditions	Person role	Person relationship	Negation	Entailment	
	83	66	66	44	33	98
2020	Conditions	Predicate argument	Negation	Legal fact	Person role	
	74	73	69	55	48	112

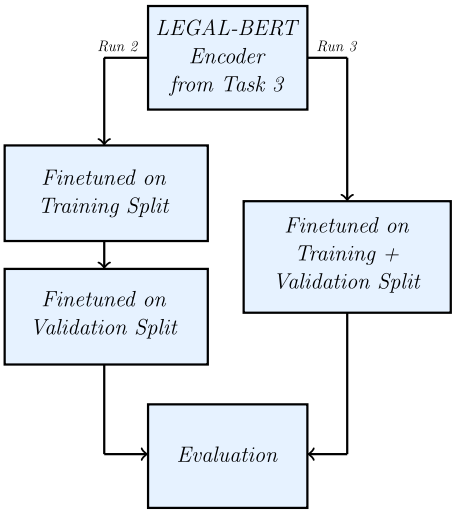

Fig. 2 Fine-tuning workflow of the LEGAL-BERT encoder for run 2 and 3

2, we fine-tune the above model on the training split and evaluate its performance on the validation set. Having achieved satisfying results, we further train the fine-tuned model on the validation split. However, for our run 3, we completely train the reinitialized classification head with the encoder base from task 3 on the entire training split including the validation split. Each run is trained using the same set of hyper-parameters that resulted in the best performance for the validation split.

To sum up, we re-used the LEGAL-BERT encoder we already had trained on the COLIEE statute law retrieval task [25] and reinitialized its classification head. To make learning easier, we decomposed the instances with multiple articles for a query, forming additional instances. Run 2 was trained in 2 steps. The first step was to fine-tune it on our training split and then in the second step, we performed further fine-tuning on the validation split. Run 3 was fine-tuned on the full training data in one step. In the next section, we evaluate all three runs.

3.3 KERMIT+BERT

In this section, we explain the architecture we used to classify textual entailment using the KERMIT encoder [30] combined with a BERT model. The idea of the KERMIT encoder is based on Recursive Neural Networks (RecNN), which process binary tree structures in the manner of a Recurrent Neural Network [24], as well as Distributed Tree Kernels [29], which encode high-dimensional tree fragments into a lower-dimensional vector representation. In particular, the tree node labels are initially one-hot encoded using a multivariate Gaussian distribution $\mathbf{v} \sim \mathcal{N}\left(0, \frac{1}{\sqrt{d}}\mathbb{I}\right)$. With the shuffled circular convolution $\mathbf{u} \otimes \mathbf{v}$ the embeddings are composed, which ensures the interpretability of the resulting vector space [29]. In combination with a BERT model, we obtain the architecture shown in Fig. 3. In our entailment scenario we generate parse trees from query and article(s) as input sequences separately, resulting in two concatenated embedding vectors in KERMIT. For the BERT model, we generate one sequence from query and article, delimited by the separation token [SEP]. Per default, BERT's [CLS] token embedding is used as a representation of the input sentence. The generated embeddings from BERT and KERMIT are then forwarded to a fully connected decoder layer with softmax activation to generate the prediction of the entailment label.

4 Evaluation

We start this section with details about the experimental setup, followed by results from the competition, previous experiments and a final discussion. For hyper-parameter optimization we evaluated our runs on the validation split of queries starting with "R01-*".

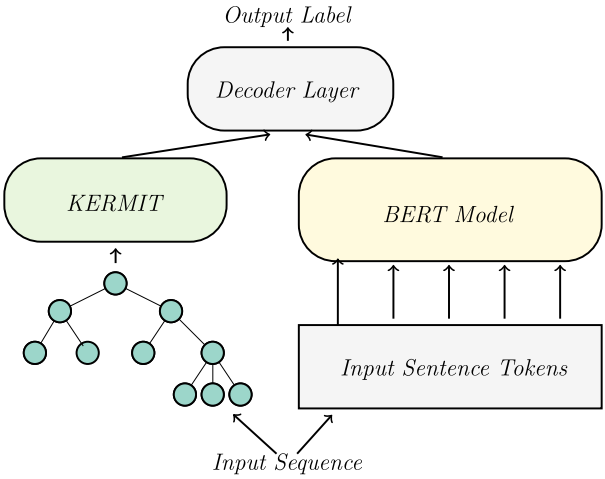

Fig. 3 KERMIT+BERT architecture, adapted from Zanzotto et al. [30]

4.1 Experimental Setup

4.1.1 Graph Neural Networks

We modeled Graph Neural Networks with different structures—words as nodes and sentences as nodes; different message passing modalities—unidirectional (from articles to query) and bidirectional (from articles to queries and vice versa). These experiments are detailed below.

Words as nodes: Initially, we constructed graphs by considering each word as a node and making connections between nodes (words) which are neighbors to that word in the sentence. We trained Word2Vec [13] vectors from the article descriptions in the Civil Code and the queries in the training dataset, using the *gensim*[5] library and used these as the features for each node. This way we tried creating graphs for both queries and articles and create a graph embedding for each graph, to combine their embeddings and use it for the entailment downstream task. But this approach did not seem to perform well on the validation split, with an accuracy of 56%.

Sentences as nodes: To improve the results, we embed the sentences of the article and the query, respectively, using Sentence-BERT and use it as a node feature. We encode this information to get query node embeddings, as described in Sect. 3. These are then passed to the linear layer. The results have been found to outperform the above-mentioned approach of considering the Word2Vec representation for the article and query token nodes.

Data enrichment with article metadata: From experimenting with different data enrichment strategies, we found that the model tends to give better results

[5] https://radimrehurek.com/gensim/models/word2vec.html.

Table 5 Influence of data enrichment on COLIEE validation set

	Correct answers	Accuracy
Without metadata	66/111	0.5946
With metadata	73/111	0.6577

Table 6 Experiments with Graph Neural Networks (with message passing from articles to query—BN and Sequence Length

Description	Validation data		COLIEE data	
	Train acc	Test acc	Train acc	Test acc
Without BN, seq=128, ew=1	0.614	0.527	0.587	0.531
With BN, seq=128, ew=1	**0.666**	**0.657**	**0.649**	0.444
With BN, seq=512, ew=1	0.652	0.500	0.643	0.531

Table 7 Experiments with Graph Neural Networks (with message passing from articles to query)—Edge Weights with BN

Edge weight	Validation data		COLIEE data	
	Train acc	Test acc	Train acc	Test acc
1	0.652	0.500	**0.643**	0.531
cosine similarity	0.652	**0.545**	0.631	**0.580**

(an improvement of over 6 percentage points in the validation data) when we add the metadata of the article while embedding it. This is shown in Table 5.

Choice of using batch normalization: Table 6 shows how the parameters affect the model performance when the message is passed from articles to query. Plugging in a batch normalization (BN) layer after each graph layer is results in better entailment classification performance. Accuracy increased significantly for all our data splits (roughly 6 percentage points), except for the COLIEE Test data where it dipped to 44% from 53%.

Choice of sequence length: While experimenting with the maximum sequence length to obtain the Sentence-BERT embeddings, it was surprising to note that even the model with a default sequence length of 128 performed reasonably well, compared to the model with a maximum sequence length of 512. This is shown in Table 6.

Choice of edge weights: We experimented with an edge weight of 1 to give equal importance to each article and also experimented with the cosine similarity between query and article as the edge weight. It is evident from Table 7 that using cosine similarity between the nodes (articles and queries) offers better performance in our setting than using a constant edge weight of 1. With both the validation and COLIEE test splits, we see this trend with an improvement of roughly 4 percentage points.

Choice of message passing modality: Though we assumed that the bidirectional message passing would fetch better results than its unidirectional counterpart, we

Table 8 Experiments with Graph Neural Networks (with message passing from query to articles and then articles to query) with BN, seq=512 and $\beta = 0.1$

Edge weight	Validation data		COLIEE data	
	Train acc	Test acc	Train acc	Test acc
1	0.633	**0.518**	0.648	**0.556**
Cosine similarity	**0.653**	0.509	**0.659**	0.481

were taken by surprise to see the former performing worse in the test splits. This is evident from Table 8.

To sum up, for the COLIEE competition, we submitted the unidirectional Graph Neural Network with sentence embeddings of sequence length 128 and an edge weight of 1 (refer to Table 6, second row). The Graph Neural Network has 2 graph layers with a batch normalization layer after each graph layer. We then add a linear layer with a rectified linear unit activation and a final linear layer with a softmax activation to perform the downstream task of entailment using PyTorch[6].

We evaluate our model on the validation split (all queries starting with the ID "R01-*"). After hyper-parameter optimization on the validation data, we train our model for 3 epochs as it was observed that the model was overfitting on the training set when the number of epochs was increased. We train our model with the Adam optimizer and with a batch size of 4 on the training split. We train the second model with the full training dataset including the validation split with the same hyper-parameters, so that the model makes use of all the data we have. To get the label predictions on the COLIEE 2021 test set, we take average softmax values from both models and choose the label with the higher confidence value.

In previous experiments, we also employed LEGAL-BERT [2] in the Sentence-BERT architecture (accuracy on the validation set: 49.55%), but the performance was not comparable to the *paraphrase-distilroberta-base-v1* model, since LEGAL-BERT was not trained with the Siamese architecture of Sentence-BERT on a natural language inference task. If there were larger legal corpora for pre-training models with the Siamese architecture, a domain-specific model may have the potential to boost the GNN performance.

4.1.2 LEGAL-BERT

For both run 2 and 3, we use the same experimental setting of hyper-parameters, albeit using different training techniques. We validate by decaying the learning rate using a decay rate of $(0.1^{(1+epoch)})$, varying the warmup steps from 5 to 20 % of the total training steps, and testing with other hyperparameters as shown in Table 9. However, warmup steps and decay rate did not have a significant impact on the performance of the LEGAL-BERT for task 4. Plausible reasons could be the small amount of training data and the fewer number of epochs used during training, resulting in a very small number of warmup steps, thereby, not providing the network

[6] https://pytorch.org/.

Table 9 Optimizing hyper-parameters on the validation set for task 4 COLIEE 2021

Correct	Epochs	Batch-size	Learning_rate	Warm_up	Decay	Accuracy
57	4	16	$1e^{-05}$	300	$0.1^{(1+\text{epoch})}$	0.5135
61	5	16	$1e^{-05}$	400	$0.1^{(1+\text{epoch})}$	0.5495
63	5	8	$1e^{-05}$	600	$0.1^{(1+\text{epoch})}$	0.5676
68	5	8	$1e^{-05}$	–	–	0.6126
71	4	8	$1e^{-05}$	–	–	0.6396
73	4	8	$5e^{-05}$	–	–	**0.6577**

Table 10 Task 4 Results for COLIEE 2021

Correct	Run	Accuracy
36	OvGU_run1	0.4444
45	OvGU_run2	0.5556
48	OvGU_run3	**0.5926**
47	KERMIT+BERT	0.5802

enough time to adapt gradually. Also, the decay rate shrinks abruptly causing no significant effect on the learning process. For this reason, we left out the warmup steps and the decay rate for training our models.

Finally, with hyper-parameter tuning, we observed that the learning rate of $5e^{-05}$ with a batch size of 8 trained for 4 epochs was the most suitable of all regarded options of our scenario.

4.1.3 KERMIT+BERT

For most part, we follow the default parameters set for the KERMIT+BERT architecture by Zanzotto et al. [30]. The dimensionality of the embedded constituency parse tree is 4000 for the tree of query and article, respectively. We obtained both trees using the same parser from Stanford Core NLP as the original authors. Then we used the Transformer model *bert-base-uncased* with 768 dimensions and a maximum sequence length of 512. The architecture is trained with the AdamW optimizer with a learning rate of $3e^{-5}$. We train the KERMIT+BERT model with a batch size of 64 for 4 epochs, more epochs led to overfitting due to the limited dataset size in COLIEE. Being aware of the effect of weight initialization in BERT models, we set multiple random seeds and train several models on the architecture.

4.2 Results

Among our submissions, OvGU_run3 achieved the fifth-highest accuracy, followed by OvGU_run2 with the sixth-highest accuracy obtaining 48 and 45 correct answers, respectively, given 81 test queries. Since both these runs are BERT-based

Table 11 Evaluation result on validation data

Method	Correct answers	Accuracy
GNN	73/111	0.6577
LEGAL-BERT	73/111	0.6577
KERMIT+BERT	58/111	0.5225

approaches, we witness comparable results. Although none of our runs could achieve the highest performance, OvGU_run1 is a novel graph-based approach, which obtained 36 correct answers. The results are populated in Table 10. Aside from the official competition results, the best performance of the KERMIT+BERT architecture on the COLIEE 2021 data is also included in the table with an accuracy of 58%. Considering the average performance of KERMIT+BERT across all 7 seeds, we obtain an average accuracy of 51.03%, with a standard deviation of 5 percentage points.

4.3 Discussion

In this section, we briefly discuss the overall results of our approaches on the COLIEE 2021 competition and point out a few discrepancies in the dataset.

4.3.1 Discussion on Task Results

We can see that our method performs well when we consider the validation data (based on queries starting with the ID "R01-*") in Table 11. Compared to this year's test set, we could observe that LEGAL-BERT was able to generalize much better than the GNN and was able to give 48 correct answers while the GNN performed worse. Graphs are usually used to represent large amounts of data, and since the dataset is limited, it may be assumed that the GNN was not able to generalize much with the dataset and the current hyperparameter settings and was not able to correctly predict the entailment relationship with the query node embeddings. We assume that graph-based techniques can achieve better scores in the future, since our approach for run 1 does not rely on much external knowledge, which could be encoded after performing more document enrichment. We also include the validation accuracy of the best-performing KERMIT+BERT model on the test data. Here, the average performance on the validation set is an accuracy of 52.4% with a standard deviation of 2.9 percentage points. Given the overall performance of the GNN and the rather low explainability of the generated embeddings, we compare it with the other approaches to find some interesting trends.

We examine all three submitted runs with the ground truth provided by the COLIEE organizers. Results confirm that our run 3 predicts more correct answers than our other two runs. Out of 81 queries, run 1 correctly predicted a total of 36 queries of which 11 queries were not correctly predicted by any of the other runs we submitted. Similarly, for run 3 out of our 48 correct predictions, 11 queries were not correctly predicted by the other runs we made. When it comes to run 2, we had

a total of 45 correct predictions where 5 queries were not correctly predicted by our other runs. For run 2 and 3, most of their predictions would have an agreement with each other, precisely 56 such queries, the reason might be since both of the runs are BERT-based approaches and even share the same initial encoder. Additionally, we observed that we could not predict 11 queries correctly in any of our runs while 16 were correctly predicted across all three. Considering only the 38 positive entailment instances from the ground truth, we characterize the model behavior based on the problem type the respective model can overcome to find a positive entailment. Since there is no official ground truth regarding the problem category assignment, we provide several examples of our approach to categorize the instances. Run 1 sometimes exclusively detected the entailment in instances with a *Condition* and *Person Role*. Example instances of this are *R02-4-I* and R02-9-O, which both contain mentions of persons with alphabetical letters (A, B,...) whose roles need to be matched to the conditions mentioned in the articles, such as being an agent concluding a sales contract. We identified Conditions by using signal words, such as "if" and "unless". Among the 28 instances with a positive entailment label that we attributed to the Condition category, run 1 identified 15 thereof, run 2 found 27, run 3 detected 16 and the KERMIT+BERT model found 10. Interestingly, run 3 detected less conditions than run 2, because run 2 overall predicted the positive entailment 54 times, whereas run 3 only 39 times, which was closer to the actual distribution in the ground truth. The same applies to the problem categories *Person Role* and *Person Relationship*, where run 2 detected 17 out of 24 instances in both categories, while all other models identified less than 10 positive entailment instances with this attribute. Run 3 was usually performing consistently well with instances with a high overlap, which not all models did effectively. Interestingly, KERMIT+BERT slightly outperformed the other models with 5 out of 8 *Paraphrases* and 6 out of 11 *Verb Paraphrases*. We assume that this architecture enables a model to align the paraphrased components and identify the entailment relationship better than other models.

We further notice that the LEGAL-BERT-based approach performs considerably better than the graph-based approach for the test queries associated with multiple articles, such as for pair IDs "R02-16-O", "R02-24-E" with an overall of 16 such queries. Our run 1, the graph-based model, is able to correctly predict 6 such queries while both BERT-based models of run 2 and 3 predicted 10 and 11 queries correctly. We believe that this is due to our input data decomposition for runs 2 and 3, recognizing each relevant article as a separate training instance. Furthermore, Sentence-BERT has a default token limit of 128, which means that all inputs with more tokens than that will be truncated. This has possibly affected the performance of run 1 for longer articles with the relevant entailment content at the end of the article part. For instance the encoding of Article 567 with 212 tokens did not change after we added the metadata, hence for articles with this length, that measure had no effect. This drawback of using the default settings shall be considered in future work with Sentence-BERT.

Apart from our submitted runs, we evaluate another run that we did not submit in the competition. This model predicts 54 correct entailment labels for 81 test queries. This submission could have been the third-highest score. Its LEGAL-BERT model

is only fine-tuned on the training split we had during validation, so that we do not use the queries starting with the ID "R01-*", as mentioned earlier. The model predicts 73 correct labels on the validation set, listed as the last entry in Table 9. We attribute this effect to the unstable training / test results on the COLIEE competition over time, such that the problem type distribution of queries starting with "R01-*" could be probably skewed. The opposite effect occurred for the KERMIT+BERT model, which has a lower validation accuracy than on the test data. Considering the relatively higher standard deviation on the test data, there seems to be a problem type distribution shift from the "R01-*" validation set and the "R02-*" test set, which may explain the large differences between the model performances.

4.3.2 Discussion on Dataset

In general, while reviewing the training and test dataset provided for COLIEE 2021 task 4, we find a few discrepancies.

1. **Incomplete article description:** For multiple queries including *H22-23-E*, *H27-23-E*, *H27-23-O*, *R01-25-U* and *R01-25-O* where *Article 617* was marked one among the relevant articles, the description in the query-article pair was incomplete in the training dataset when compared with the article description in the Civil Code.
2. **Mentioning only selected paragraphs of the article:** In the training dataset, *Article 718* was marked as relevant for the query *H30-29-E*. Though the article had 2 different paragraphs in the Civil Code, only the 1st paragraph was provided in the query-article pair in the training dataset.

   This can be neglected owing to the count of 1 on a total of 806 train instances, however, when we consider the test dataset, there are 35 such instances on a total of 81.

   This is particularly interesting when different paragraphs of the same article were mentioned for different queries. For query *R02-27-A*, only the second paragraph of *Article 676* is mentioned while for query *R02-27-O*, the third paragraph is mentioned. This brings us to the question if the selected mentioning was intentional. In this case, task 4 could extend its scope to not just checking for an entailment label, given relevant articles and a query, but also to find the most relevant paragraph(s) in the given article and then check for an entailment label with the query.
3. **Mismatch in article number and article description:** For the test query *R02-2-E*, the description of the relevant article seems to be incorrect. When comparing with the Civil Code, the article description of *Article 36* was used in the test query-article pair, but the article number mentioned was *Article 35*. It is also interesting to note the original article description for *Article 35* in the Civil Code has Japanese characters.
4. **Article description in Japanese:** The article description for the test query (in the English test dataset) *R02-5-I* is given in Japanese. This puts the competitors

using the English dataset at a disadvantage of potentially losing one test instance and thereby decreasing the number of correct predictions.

From the discussions and findings above, our main takeaways from COLIEE'21 task 4 are:

1. Data decomposition of queries associated with multiple articles into multiple instances can help the neural networks to model the query-article relation better.
2. Developing an understanding of the data distribution or the skewness in individual training dataset subsets ("H28-*", "H29-*", "R01-*", and more) can help to address the queries better. It would be interesting to have an official multi-label assignment of problem categories released for the training data.

5 Conclusion and Future Work

To conclude, we test graph and contextual word embeddings for task 4 of textual entailment classification in the COLIEE competition. In particular, we use Graph Neural Networks with sentence embeddings in run 1 and LEGAL-BERT variations in training, with two stages in run 2 and one stage for run 3. We find that increasing the number of instances with article decomposition can help to boost the performance of our approaches. From all submitted runs, the LEGAL-BERT run 3 which was fine-tuned on all available data in one stage performed best. However, we found that the training / test split can substantially impact the model performance, which is shown by run 2, 3 and an auxiliary run which outperformed all submitted runs. In addition to the submitted competition runs we experimented with the KERMIT+BERT architecture to encode syntactic parse trees and use them jointly with contextual embeddings. Initial results for the *bert-base-uncased* model are promising and may improve with domain-specific pre-trained transformers in follow-up experiments. For future work, we will test KERMIT's visualization component for the heat parse trees and intend to focus more on the generation of sentence embeddings and their appropriate aggregation for longer articles. Furthermore, we suggest to incorporate external knowledge via more extensive document enrichment with knowledge from the web into all three approaches as an addition to the decomposition and augmentation strategies we employed this time.

Funding Open Access funding enabled and organized by Projekt DEAL.

Declarations

Conflict of interest The authors declare that they have no conflict of interest.

References

1. Cao, N. D., Aziz, W., & Titov, I. (2019). Question answering by reasoning across documents with graph convolutional networks. In: Burstein, J., Doran, C., Solorio, T. (eds.), Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers). pp. 2306–2317. Association for Computational Linguistics
2. Chalkidis, I., et al. (2020). LEGAL-BERT: the muppets straight out of law school. CoRR arxiv:2010.02559
3. Clark, K., Luong, M., Manning, C.D., & Le, Q.V. (2018). Semi-supervised sequence modeling with cross-view training. In: Riloff, E., Chiang, D., Hockenmaier, J., Tsujii, J. (Eds.), Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31—November 4, 2018. pp. 1914–1925. Association for Computational Linguistics. https://doi.org/10.18653/v1/d18-1217
4. Devlin, J., Chang, M., Lee, K., & Toutanova, K. (2019). BERT: pre-training of deep bidirectional transformers for language understanding. In: Burstein, J., Doran, C., Solorio, T. (Eds.), Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2–7, 2019, Volume 1 (Long and Short Papers). pp. 4171–4186. Association for Computational Linguistics https://doi.org/10.18653/v1/n19-1423
5. Gilmer, J., Schoenholz, S.S., Riley, P.F., Vinyals, O., & Dahl, G.E. (2017). Neural message passing for quantum chemistry. In: Precup, D., Teh, Y.W. (Eds.), Proceedings of the 34th international conference on machine learning, ICML 2017, Sydney, NSW, Australia, 6–11 August 2017. Proceedings of Machine Learning Research, vol. 70, pp. 1263–1272. PMLR. http://proceedings.mlr.press/v70/gilmer17a.html
6. Gu, Y., et al. (2020). Domain-specific language model pretraining for biomedical natural language processing. CoRR. arxiv:2007.15779
7. He, C., et al. (2020). Cascade-bgnn: Toward efficient self-supervised representation learning on large-scale bipartite graphs
8. Hirschberg, J., & Manning, C. D. (2015). Advances in natural language processing. *Science, 349*(6245), 261–266.
9. Hoshino, R., et al. (2018). Question answering system for legal bar examination using predicate argument structure. In: Kojima, K., Sakamoto, M., Mineshima, K., Satoh, K. (Eds.), New Frontiers in Artificial Intelligence—JSAI-isAI 2018 Workshops, JURISIN, AI-Biz, SKL, LENLS, IDAA, Yokohama, Japan, November 12–14, 2018, Revised Selected Papers. Lecture Notes in Computer Science, vol. 11717, pp. 207–220. Springer
10. Howard, J., & Ruder, S. (2018). Universal language model fine-tuning for text classification. In: Gurevych, I., Miyao, Y. (Eds.), Proceedings of the 56th annual meeting of the association for computational linguistics, ACL 2018, Melbourne, Australia, July 15–20, 2018, Volume 1: Long Papers. pp. 328–339. Association for Computational Linguistics. https://www.aclweb.org/anthology/P18-1031/
11. Kipf, T.N., & Welling, M. (2016). Semi-supervised classification with graph convolutional networks. arXiv preprint arXiv:1609.02907
12. Kuncoro, A., Kong, L., Fried, D., Yogatama, D., Rimell, L., Dyer, C., & Blunsom, P. (2020). Syntactic structure distillation pretraining for bidirectional encoders. *Transactions of the Association for Computational Linguistics, 8*, 776–794. https://doi.org/10.1162/tacl_a_00345
13. Kusner, M.J., Sun, Y., Kolkin, N.I., & Weinberger, K.Q. (2015). From word embeddings to document distances. In: Bach, F.R., Blei, D.M. (Eds.), Proceedings of the 32nd international conference

on machine learning, ICML 2015, Lille, France, 6–11 July 2015. JMLR workshop and conference proceedings, vol. 37, pp. 957–966. JMLR.org
14. Lan, Z., et al. (2020). ALBERT: a lite BERT for self-supervised learning of language representations. In: 8th International conference on learning representations, ICLR 2020, Addis Ababa, Ethiopia, April 26–30, 2020. OpenReview.net. https://openreview.net/forum?id=H1eA7AEtvS
15. Liu, Y., et al. (2019). Roberta: a robustly optimized BERT pretraining approach. CoRR arxiv:1907.11692
16. McCann, B., Bradbury, J., Xiong, C., & Socher, R. (2017). Learned in translation: Contextualized word vectors. In: Guyon, I., von Luxburg, U., Bengio, S., Wallach, H.M., Fergus, R., Vishwanathan, S.V.N., Garnett, R. (Eds.), Advances in Neural Information Processing Systems 30: Annual conference on neural information processing systems 2017, December 4–9, 2017, Long Beach, CA, USA. pp. 6294–6305
17. Morris, C., et al. (2019). Weisfeiler and leman go neural: Higher-order graph neural networks. In: The Thirty-Third AAAI conference on artificial intelligence, AAAI 2019, The thirty-first innovative applications of artificial intelligence conference, IAAI 2019, The Ninth AAAI symposium on educational advances in artificial intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27—February 1, 2019. pp. 4602–4609. AAAI Press
18. Nguyen, H., et al. (2020). JNLP team: Deep learning for legal processing in COLIEE 2020. CoRR arxiv:2011.08071
19. Peters, M.E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., & Zettlemoyer, L. (2018). Deep contextualized word representations. In: Walker, M.A., Ji, H., Stent, A. (eds.) Proceedings of the 2018 conference of the North American chapter of the association for computational linguistics: human language technologies, NAACL-HLT 2018, New Orleans, Louisiana, USA, June 1–6, 2018, Volume 1 (Long Papers). pp. 2227–2237. Association for Computational Linguistics, https://doi.org/10.18653/v1/n18-1202
20. Rabelo, J., Kim, M.Y., Goebel, R., Yoshioka, M., Kano, Y., & Satoh, K. (2020a). Coliee 2020: methods for legal document retrieval and entailment. https://sites.ualberta.ca/~rabelo/COLIEE2021/COLIEE2020_summary.pdf. Accessed date 09 May 2021
21. Rabelo, J., et al. (2020b). A summary of the coliee 2019 competition. https://sites.ualberta.ca/~rabelo/COLIEE2020/COLIEE2019_summary.pdf. Accessed date 09 May 2021
22. Radford, A., Narasimhan, K., Salimans, T., & Sutskever, I. (2018). Improving language understanding by generative pre-training. https://www.cs.ubc.ca/~amuham01/LING530/papers/radford2018improving.pdf, note=. Accessed date 09 May 2021
23. Reimers, N., & Gurevych, I. (2019). Sentence-bert: Sentence embeddings using siamese bert-networks. In: Inui, K., Jiang, J., Ng, V., Wan, X. (Eds.), Proceedings of the 2019 conference on empirical methods in natural language processing and the 9th international joint conference on natural language processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3–7, 2019. pp. 3980–3990. Association for Computational Linguistics
24. Socher, R., Lin, C.C., Ng, A.Y., & Manning, C.D. (2011). Parsing natural scenes and natural language with recursive neural networks. In: Getoor, L., Scheffer, T. (Eds.), Proceedings of the 28th international conference on machine learning, ICML 2011, Bellevue, Washington, USA, June 28–July 2, 2011. pp. 129–136. Omnipress. https://icml.cc/2011/papers/125_icmlpaper.pdf
25. Wehnert, S., et al. (2021). Legal norm retrieval with variations of the bert model combined with tf-idf vectorization. In: Proceedings of the eighteenth international conference on artificial intelligence and law. p. 285–294. Association for Computing Machinery, New York, NY, USA
26. Xu, Y., & Yang, J. (2019). Look again at the syntax: Relational graph convolutional network for gendered ambiguous pronoun resolution. CoRR arxiv:1905.08868
27. Yao, L., Mao, C., & Luo, Y. (2019). Graph convolutional networks for text classification. In: The thirty-third AAAI conference on artificial intelligence, AAAI 2019, the thirty-first innovative applications of artificial intelligence conference, IAAI 2019, The ninth AAAI symposium on educational advances in artificial intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27–February 1, 2019. pp. 7370–7377. AAAI Press, https://doi.org/10.1609/aaai.v33i01.33017370
28. Yoshioka, M., et al. (2020). Overview of japanese statute law retrieval and entailment task at coliee-2018. https://sites.ualberta.ca/~rabelo/COLIEE2019/COLIEE2018_SL_summary.pdf. Accessed date 09 May 2021
29. Zanzotto, F.M., & Dell'Arciprete, L. (2012). Distributed tree kernels. In: Proceedings of the 29th international conference on machine learning, ICML 2012, Edinburgh, Scotland, UK, June 26–July 1, 2012. icml.cc / Omnipress. http://icml.cc/2012/papers/111.pdf

30. Zanzotto, F.M., Santilli, A., Ranaldi, L., Onorati, D., Tommasino, P., & Fallucchi, F. (2020). KERMIT: complementing transformer architectures with encoders of explicit syntactic interpretations. In: Webber, B., Cohn, T., He, Y., Liu, Y. (Eds.), Proceedings of the 2020 conference on empirical methods in natural language processing, EMNLP 2020, Online, November 16–20, 2020. pp. 256–267. Association for Computational Linguistics. https://doi.org/10.18653/v1/2020.emnlp-main.18