EPiC Series in Computing

Volume 47, 2017, Pages 31–42

COLIEE 2017. Fourth Competition on Legal Information Extraction/Entailment

Recognizing entailments in legal texts using sentence encoding-based and decomposable attention models

Nguyen Truong Son, Phan Viet Anh, and Nguyen Le Minh

Japan Advanced Institute of Science and Technology, Nomi, Ishikawa, Japan 923-1292
`nguyen.son@jaist.ac.jp, anhpv@@jaist.ac.jp, nguyenml@jaist.ac.jp`

Abstract

This paper presents an end-to-end question answering system for legal texts. This system includes two main phases. In the first phase, our system will retrieve articles from Japanese Civil Code that are relevant with the given question using the cosine distance after the given question and articles are converted into vectors using TF-IDF weighting scheme. Then, a ranking model can be applied to re-rank these retrieved articles using a learning to rank algorithm and annotated corpus. In the second phase, we adapted two deep learning models, which has been proposed for the Natural language inference task, to check the entailment relationship between a question and its related articles including a sentence encoding-based model and a decomposable attention model. Experimental results show that our approaches can be a promising approach for information extraction/entailment in legal texts.

Keywords: legal question answering, information retrieval, entailment extraction, deep learning

1 Introduction

Studying legal issues from the perspective of informatics is a topic that draws interests from researchers recent years. In this task, one of the important targets is the information extraction/reasoning from legal data including legal relation extraction, textual entailment, etc. Legal question answering is also an essential task in legal issues. Recent work proposed some of the first efforts for this task [8, 15, 6].

In this work, we build a two-phase system for the legal information extraction/entailment task. In the first phase, a list of relevant articles are retrieved by compute the cosine similarity between the TF-IDF vectors of the given question and articles. A ranking model can be applied to obtain the best results. The architecture of this phase is followed the architecture of our system last year [12] by adding some n-gram indexing models ($n = 2$) beside the uni-gram indexing model.

A question in the COLIEE task is a statement that we need to check whether or not it is entailed by some articles in legal documents. Therefore, the second phase will try to check entailment relationship between a question and its related articles thanks to a classifier. Because we have an annotated data set, supervised methods are preferred approaches for this task by treating this task as a binary classification task.

Recently, deep learning models show it effectiveness for NLP tasks such as machine translation, sequence labeling, text generation as well as natural language inference task. In many tasks such as named entity recognition, deep learning models can produce the state-of-the-art without using any engineering features [9]. In the natural language inference tasks [2], which try to recognize the entailment relationship between a text and a hypothesis, deep learning models also produce better results in comparison with conventional machine learning algorithms [2]. For these reasons, our motivation is how to adapt these models for question answering task in legal texts. In this work, we investigate two deep learning models including the sentence encoding-based model proposed by [2], and the decomposable attention model proposed by [13].

Experimental results show that our approaches have some promising results. First, adding bi-gram and tri-gram indexing models shows a significant improvement. Second, deep learning models could be a good approaches for entailment recognition in legal texts because without engineering features, the result is competitive with conventional algorithms.

The remainder of this paper is organized as follows. Section 2 presents briefly some related works. The architecture of Information retrieval phase and entailment checking phase are presented in Section 3 and 4. The experiments are analyzed in Section 5. Conclusions and future works are shown in Section 6.

2 Related Work

A basis for a Yes/No Arabic Question Answering System was proposed in [1] using a textual entailment method. For the task of legal information retrieval/entailment task, [8] proposed a system for answering yes/no questions in legal bar exams.

The first shared task of the legal information retrieval/extraction (COLIEE2014) was reported in [15]. There are many methods proposed to solve the task. [17] focused on exploiting reference information to build a question answering system restricted to the legal domain. [7] used ranking SVM and syntactic/semantic similarity.

In the COLIEE 2015 competition, [6] proposed using a convolutional neural network in legal question answering. D. S. Carvalho et al. [3] used lexical and morphological characteristics to extract n-gram features from the query and articles; their own relevance score was then used for Phase 1. They used the query and relevant article are represented in vector space model for Phase 2. In order to classify the queries between Yes label and No label, they used AdaBoost (basic classifier: DecisionStump). Sushimita et al. [14] used voting of three information retrieval techniques: Hiemstra, BM25 and PL2F for the information retrieval task. In [4], a keyword weighting method was proposed and snippet scoring after diving data into snippets. Tran et al., 2015 [16] proposed using hidden markov model for the legal information retrieval task.

In the COLIEE 2016, [5] proposed ensemble similarity using a least square method and linear discriminant analysis with variety of features such as lexical similarity, syntactic similarity, and semantic similarity. Among their many approaches, LSM ensemble showed best performance. Besides, for the entailment recognition task, they applied majority voting scheme with three classifiers (decision tree, linear SVM, and convolutional neural network (CNN) classifiers) using many features such as word overlap, cosine similarity, substring similarity, the Lucene similarity score, 3 different role similarity scores, 6 WordNet similarity scores, negative term ratio, and length ratio. Their system showed the best performance in COLIEE 2016.

3 Phase 1: Retrieving relevant articles

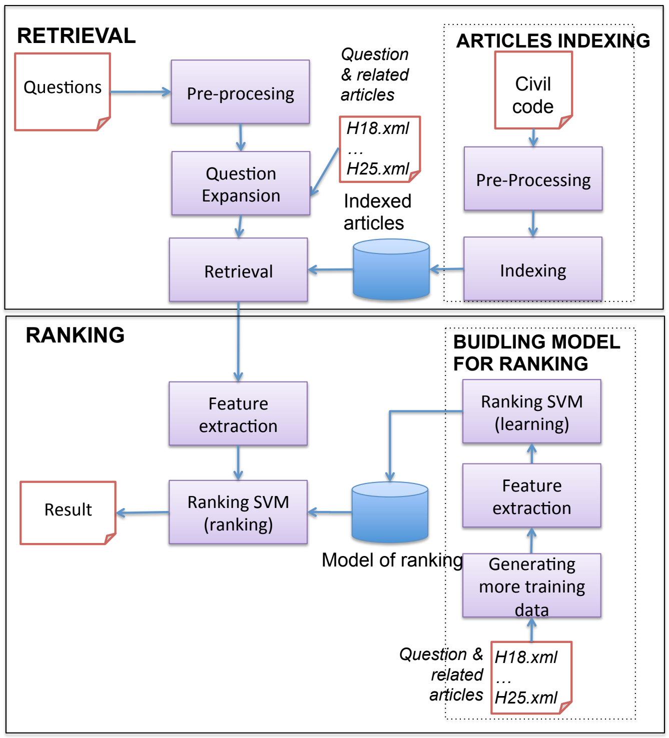

Figure 1: The architecture of information retrieval phase used in [12]

The architecture of the information retrieval system is described in Figure 1 which is followed by the architecture in [12]. It has two main steps including information retrieval and ranking. In the first step, for each given question, the system will retrieve top 100 relevant articles based on the cosine distance between the TF-IDF vectors of articles and the given question.

The ranking step will re-rank 100 relevant articles of the first step by using a ranking model which is trained by SVM rank algorithm. Finally, for each question we choose the most relevant article after the ranking step.

The details of these steps will be described in the remain of this section.

3.1 Articles Indexing

This component will convert all articles into TF-IDF vector representations, a popular method for representing documents in the information retrieval field. Moreover, to improve the performance of the system, some preprocessing steps can be applied such as stemming or removing stop words. Besides, to increase the important of long text matching, instead of using only uni-gram model, we add bi-gram and tri-gram indexing model for documents representation.

3.2 Query Expansion

The query expansion step is an option in our system. This step tries to add related terms into a given query to improve retrieve performance. Query expansion involves techniques that can find synonyms of words, and search for the synonyms as well. We used two methods for query expansion using Word2Vec and WordNet.

Query expansion using word2vec[10]: From questions and their relevant articles in training corpus, extract similar word pairs of word in question and articles base on cosine similarity of their word embedding representation. We select word-pairs that have the cosine similarly value ≥ 0.5 as the related words for query expansion. We expect this methods can retrieve articles that does not share words with the given query.

Query expansion using WordNet[11]: From questions, we expand the question by adding synonyms and hypernyms of words in that question by looking in WordNet dictionary. However, this way is not effective because each word may have it make the precision score reduce sharply.

3.3 Ranking

3.3.1 Prepare training data

Given a question, learning to rank algorithms require relevant articles and non-relevant articles for that question as the training data to build the model for ranking. However, the COLIEE training data only contains questions and relevant articles, so it is not enough for learning to rank algorithm work effective. To get more training data for learning to rank algorithm, we generate non-relevant articles of a question by selecting articles that have the low cosine similarity value based on TF-IDF vectors.

3.3.2 Features Extraction

We used the following feature list to train the ranking model.

- Cosine similarity between question and article of TF-IDF unigram vectors
- Cosine similarity between questions and articles of TF-IDF bigram vectors
- Cosine similarity between question and article of TF-IDF trigram vectors
- Cosine similarity between nouns in questions and articles
- Cosine similarity between verbs in questions and articles
- Distance feature between questions and article: Euclidean, Manhattan, Levenshtein, Jaccard

3.3.3 Training the ranking model

We train the ranking model using SVMRank tool [1] based on the above feature set. Many previous works showed that SVM Rank are an effective tools for training ranking models. [7] also used SVM-Rank for ranking the result after the retrieval phase in the COLIEE task.

[1]https://www.cs.cornell.edu/people/tj/svm_light/svm_rank.html

4 Recognizing Entailment between articles and queries

Given a pair of text including a question **b** and a text **a** in which **a** is the content of some articles that relevant to question **b** which has been retrieved from the phase 1. This task tries to recognize whether or not **b** is entailed by **a** based on the meaning of the given question and related articles. By considering the content of related articles as a **text**, and the content of the given question is **hypothesis** we can solve this task using some approaches for the natural language inference task that have been developed in recent years.

We adapt two types of deep learning models to recognize the entailment relationship between law articles and questions including: (1) a sentence encoding based model that uses recurrent neural networks to encode the content of an article and the given question; and (2) a decomposable attention model which try to solve the textual entailment task by computing the alignment between words and phrases of the given question and related articles then use these alignments as features to check the entailment relationship.

Training data set: each training example in the training set is a triple of $\{\mathbf{a}^{(n)}, \mathbf{b}^{(n)}, \mathbf{y}^{(n)}\}$ where $\mathbf{a}^{(n)}$, $\mathbf{b}^{(n)}$, and $\mathbf{y}^{(n)}$ is content of related articles, a question and the label of the this training example ($\mathbf{y} \in \{Y, N\}$).

4.1 A Sentence encoding model based on Recurrent neural networks

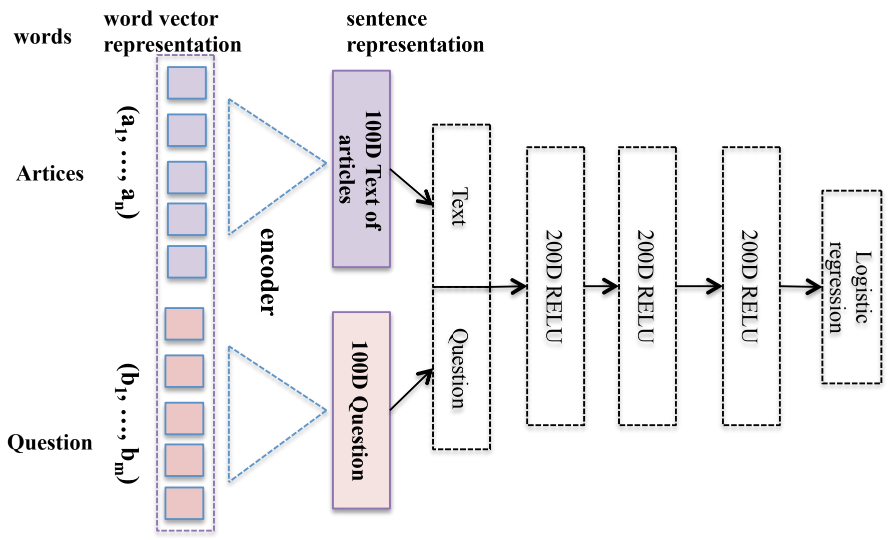

Figure 2: The sentence encoding model for recognizing the entailment between a question and the relevant articles

Figure 2 shows the architecture of the sentence encoding-based model. Let $\mathbf{a} = (a_1, a_2, ..., a_{l_a})$ and $\mathbf{b} = (b_1, b_2, ..., b_{l_b})$ be the two input texts (related articles and question) respectively. Each a_i, b_j is a word embedding vector of dimension d. l_a and l_b is the length of **a** and **b**. Then these texts are encoded into vectors (the translation step) using a sentence encoder model. After that, two these vectors are concatenated into only one vector and then this vector is fed through three layers with RELU activation. Finally, the logistic regression layer will classify the input sentence pair to predict whether or not the given article entails the question.

Below are some simple sentence encoder models which have used in [2]:

- Summation of embedding words: This method will sum the embeddings of the words in each text to obtain the vector representation.
- Simple sequence embedding models: using simple RNN or Long-short term memory to encode the input sentence.

4.2 Decomposable attention models

Figure 3 shows the architecture of the decomposable attention model for natural language inference proposed by [13] which is used to recognize the relationship between two input texts $\mathbf{a} = (a_1, a_2, ..., a_{l_a})$ and $\mathbf{b} = (b_1, b_2, ..., b_{l_b})$ by decomposing the problem into sub-problems. This model is composed from three main components: **attend**, **compare** and **aggregate**.

Attend This component will soft-align the elements of $\mathbf{a}$ and $\mathbf{b}$ using the neural attention technique. For each word a_i in $\mathbf{a}$, this step will find a sub-phrase β_i in $\mathbf{b}$ that soft-aligned to a_i and vice versa for α_j. The values of β_i and α_j are computed by equation 1 and 2.

$$\beta_i := \sum_{j=1}^{l_b} \frac{exp(e_{ij}))}{\sum_{k=1}^{l_b} exp(e_{ik})} b_j \tag{1}$$

$$\alpha_j := \sum_{i=1}^{l_a} \frac{exp(e_{ij}))}{\sum_{k=1}^{l_a} exp(e_{ik})} a_i \tag{2}$$

where e_{ij} are attention weights between words in $\mathbf{a}$ and $\mathbf{b}$ which is computed using a feed forward neural network F:

$$e_{ij} = F(a_i)^T F(b_j) \tag{3}$$

The obtained aligned phrases $\{(a_i, \beta_i)\}_{i=1}^{l_a}$ and $\{(b_j, \alpha_j)\}_{j=1}^{l_b}$ allow the model to decompose the problem into the comparison of aligned sub-phrases.

Compare This step will compare aligned phrases obtained from the previous step using a function G.

$$\begin{aligned} \mathbf{v}_{1,i} &:= G([a_i, \beta_i]) \quad \forall i \in [1, ..., l_a] \\ \mathbf{v}_{2,j} &:= G([b_j, \alpha_j]) \quad \forall j \in [1, ..., l_b] \end{aligned} \tag{4}$$

where the brackets $[\bullet, \bullet]$ denote the concatenation between two vectors. G is again a feed forward neural network with RELU activations.

Aggregate Two sets of comparison vectors $\{\mathbf{v}_{1,i}\}_{i=1}^{l_a}$ and $\{\mathbf{v}_{2,j}\}_{j=1}^{l_b}$ are aggregated into two vector $\mathbf{v}_1$ and $\mathbf{v}_2$ using summation:

$$\mathbf{v}_1 = \sum_{i=1}^{l_a} \mathbf{v}_{1,i} \qquad \mathbf{v}_2 = \sum_{j=1}^{l_b} \mathbf{v}_{1,i} \tag{5}$$

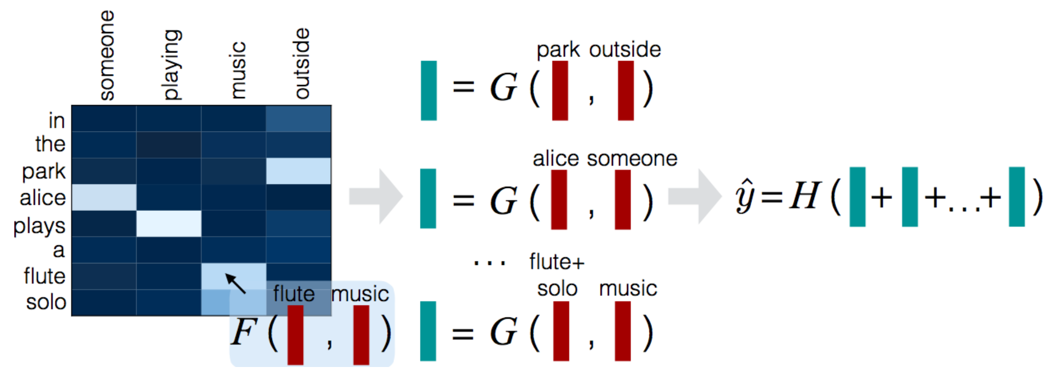

Figure 3: The decomposable attention model for recognizing textual entailment between two sentences including 3 steps: *Attend* (left), *Compare* (center) and *Aggregate* (right) [13]

Finally, a classifier H is used to predict the scores for each class. H is a feed forward network followed by a linear layer:

$$\hat{y} = H([\mathbf{v}_1, \mathbf{v}_2]) \tag{6}$$

where $\hat{y} \in \mathbb{R}^C$ is a vector that represents the scores of each classes (e.g. Yes/No). Then, the predicted class is computed by $\mathbf{argmax}_i \hat{y}_i$.

For training, the model is trained using dropout regularization with the cross-entropy loss function. During the training process, parameters of F, G, H is updated to minimize the loss function.

5 Experiments

5.1 Experimental results of phase 1

For phase one, we analysed aspects of the information retrieval task in our system including: stemming, query expansion, n-gram indexing, removing stop-words, and ranking. We conducted experiments and compared different configurations to find the configuration that produces the best performance. Table 1 shows the results of the information retrieval phase.

The results on the H18-H25 shows the contribution of the stemming and removing stop words step. When we stem or remove stop words, the performance of the system is usually better. However, these steps have negative impact on the data set H26 and H28.

Adding bi-gram indexing and tri-gram indexing models also improves the performance of our system. On all three data sets, top 3 best results are always using bi-gram or tri-gram indexing model. Experimental results in Table 2 also show that adding 2-gram and 3-gram have an important contribution for the retrieval task. For example, the retrieval performance improves 1.93%, 4.42%, 7.44% on H18-H25, H26, H28 data sets if we use the 2-gram indexing model.

However, when we use the n-gram indexing model with $n > 3$, the results do not improve but it takes more time for retrieving as well as for indexing.

Table 1 also shows that our query expansion approach does not always improve the results. Using word embedding similarity can find useful terms to the given question for but it may add many unrelated terms.

#	QUERY-EXPANSION	N-GRAM	REMOVE-STOPWORD	STEM	H18-H25	H26	H28
1		1gram			0.5096	0.5752	0.5319
2		1gram		x	0.5203	0.5487	0.5000
3		1gram	x		0.5075	0.5221	0.5532
4		1gram	x	x	0.5139	0.5133	0.4894
5		2gram			0.4989	0.5398	0.5426
6		2gram		x	0.5139	0.5487	0.5319
7		2gram	x		0.5096	0.5487	0.5638
8		2gram	x	x	0.5332	0.5575	0.5638
9		3gram			0.4968	0.5575	0.5532
10		3gram		x	0.5139	0.5752	0.5426
11		3gram	x		0.5161	0.5841	0.5638
12		3gram	x	x	0.5289	0.5664	**0.6277**$^{\textbf{(H28a)}}$
13		4gram			0.4946	**0.6018**	**0.5638**$^{\textbf{(H28b)}}$
14		4gram		x	0.5032	0.5929	0.5319
15		4gram	x		0.5139	0.5664	0.5638
16		4gram	x	x	0.5246	0.5664	0.5957
17	x	1gram			0.5203	0.5752	0.5213
18	x	1gram		x	0.5118	0.5310	0.5106
19	x	1gram	x		0.5139	0.5133	0.5426
20	x	1gram	x	x	0.5032	0.4956	**0.5213**$^{\textbf{(H28c)}}$
21	x	2gram			0.5075	0.5487	0.5745
22	x	2gram		x	0.5310	0.5487	0.5319
23	x	2gram	x		0.5246	0.5487	0.5532
24	x	2gram	x	x	**0.5439**	0.5310	**0.5851**$^{\textbf{(H28d)}}$
25	x	3gram			0.5032	0.5664	0.5532
26	x	3gram		x	0.5203	0.5752	0.5532
27	x	3gram	x		0.5246	0.5664	0.5532
28	x	3gram	x	x	0.5353	0.5664	0.5957
29	x	4gram			0.5011	0.5841	0.5638
30	x	4gram		x	0.5075	0.5841	0.5319
31	x	4gram	x		0.5268	0.5664	0.5426
32	x	4gram	x	x	0.5268	0.5575	0.5957

Table 1: Experimental results ($F_{\beta=1}$ score) of phase 1 - Information Retrieval without applying re-ranking models. H28b, H28d is the result on the test set when the performance is the best on the development set (H26) and training set respectively; H28a indicates the performance is the best on the test set but training set and development set; H28c indicates the result which we have submitted for the final official evaluation.

Table 3 shows the results after using the ranking model which is trained on the H18-H25 data set. Firstly, We retrieve relevant articles for each configuration then apply the ranking model to re-rank the retrieve articles. Experimental results show that the ranking model produces an intermediate result in comparison with the highest and the lowest results. For example, in the H26 data set, the $F_{\beta=1}$ score range is from 0.4956 to 0.6018 depend on the chosen configuration. However, after applying the ranking model, the results are 0.5221 of $F_{\beta=1}$ for all configurations.

#	N-GRAM	H18-H25	H26	H28
4	1 gram	0.5139	0.5133	0.4894
8	2 gram	0.5332	0.5575	0.5638
12	3 gram	0.5289	0.5664	0.6277
16	4 gram	0.5246	0.5664	0.5957

Table 2: Comparison between difference n-gram indexing models (all other configurations are the same: Query Expansion:No, Remove Stop words: Yes, Stemming: Yes)

This shows that the ranking model does not produce the best result.

Question set	Before ranking	After ranking
H26	0.4956∼0.6018	0.5221
H28	0.4894∼0.6277	0.5106

Table 3: Results ($F_{\beta=1}$ score) after using SVM rank model

5.2 Experimental results of phase 2

We use two available implementations [2] [3] for both two models with some adaptation for processing the input data. Both of the two tools are implemented in python language using Tensor Flow and Theano library.

Experimental settings:

- Configurations for the sentence encoding model: We experimented with two methods for sentence encoding including Word-summation and BI-LSTM. Other parameters are fixed such as word embedding size = 100, hidden layer size = 100, batch size = 8, drop-out rate = 0.2. This model is trained with training with RMSP optimizer and the training process will be stopped if the loss on the development set does not reduce after 50 epochs.
- Decomposable attention model: word embedding size=100, each feed forward network has 2 layers with hidden layer size = 200 , batch size = 8. This model is trained with Adagrad optimizer and learning rate = 0.05.

Pre-trained word embedding vectors for legal domain To train word embedding vectors for legal domain, we make a legal corpus from legal documents that crawled from website of Ministry of Justice, Japan [4]. Then, word embedding vectors are trained by using word2vec tool [10]. These embedding vectors are used for both two deep learning models in this task.

[2]https://github.com/Smerity/keras_snli
[3]https://github.com/erickrf/multiffn-nli
[4]http://www.japaneselawtranslation.go.jp: This website contain the English translation of Japanese legal documents

Results Table 4 and 5 show the results of the entailment classification task. The decomposable attention model outperforms the sentence encoding based model. Table 5 shows that the decomposable attention model usually produces the good result after 100 epoch training.

In the sentence encoding based model, using a complex network to encode input sentences does not improve the performance. For example, in 4 of 5 cases in Table 4, the BI-LSTM encoder shows lower results than word-summation encoder.

Sometimes, the model achieves the best results on the development set but it shows the bad results on the test set. It makes the parameter tuning process more difficult. We think that the size of training data and the difference between data sets are the reasons for these unstable results.

Sentence encoding method	Development set H26	H28a	H28b	H28c	H28d
Word-summation	0.6000	0.4872	0.4615	**0.5128** ♣	0.4872
BI-LSTM	0.5895	0.4359	0.4872	0.4359	0.4359

Table 4: Experimental results (accuracy) using sentence encoding-based models. H28a, H28b, H28c, H28d are obtained from phase 1 with configuration 12, 13, 20, 24 respectively. (♣) is the results we have summited for official evaluation

Epoch	Development set H26	H28a	H28b	H28c	H28d
10	0.4737	0.4615	0.4615	0.4615	0.4615
20	0.4211	0.5256	0.5256	0.4872	0.5000
30	0.4632	0.4487	0.4872	0.4359	0.4487
40	0.5895	0.4615	0.4487	0.4872	0.4872
50	0.4737	0.4359	0.4359	0.4359	0.4359
60	0.4842	0.4744	0.4615	0.4744	0.4872
70	0.5895	0.5128	0.5385	0.5513	0.5385
80	0.6000	0.5256	0.5000	0.5641	0.4744
90	0.5895	0.5641	0.5256	0.5897(*)	0.5385
100	**0.6211**	**0.5769**	**0.5256**	**0.4872**♣	**0.5128**
110	0.4526	0.5641	0.5513(*)	0.5513	0.5385
120	0.5053	0.5897(*)	0.5256	0.5385	0.5513(*)
130	0.6000	0.4231	0.4615	0.4359	0.4231
140	0.4632	0.4872	0.5128	0.5769	0.5256
150	0.4316	0.5385	0.5385	0.4872	0.5385

Table 5: Accuracy on the development set and test set of the decomposable attention model during the training process. The **bold line** indicates the results on test sets when the performance is the best on the development set. H28a, H28b, H28c, H28d are obtained from phase 1 with configuration 12, 13, 20, 24 respectively. The (*) symbols indicate the result are the best on the test set but the development set. Network setting={dropout(keep):0.8, learning rate:0.05, number node in hidden layers:200, input embedding size: 100}. (♣) is the results we have summited for official evaluation

6 Conclusion

We built a system for the legal information retrieval/entailment task. For the first phase, the most relevant article given a question was obtained based on two steps. A set of relevant articles were extracted using cosine TF-IDF vectors with some additional techniques to improve the performance such as query expansion with word embedding sources. Then, retrieved articles were re-ranked based on SVM rank to achieve the most relevant article. For the second phase, we employed two deep learning models to recognize entailments between articles and questions. One of limitation of this approach is that the size of training corpus is too small to train a stable model. In future, incorporating engineering features can be good ways to improve the performance of the system.

7 Acknowledgments

This work was supported by JAIST CREST, Japan.

References

[1] Wafa N Bdour and Natheer K Gharaibeh. Development of yes/no arabic question answering system. *International Journal of Artificial Intelligence and Applications*, 4(1):51–63, 2013.

[2] Samuel R Bowman, Gabor Angeli, Christopher Potts, and Christopher D Manning. A large annotated corpus for learning natural language inference. *arXiv preprint arXiv:1508.05326*, 2015.

[3] Danilo Carvarlho, Minh-Tien Nguyen, Chien Tran, and Le-Minh Nguyen. Lexical-morphological modeling for legal text analysis. In *Lecture notes in computer science:New Frontiers in Artificial Intelligence*. Springer, 2016.

[4] Y. Kano. Keyword and snippet based yes/no question answering system for coliee 2015. In *Ninth International Workshop on Juris-informatics (JURISIN)*, 2015.

[5] Kiyoun Kim, Seongwan Heo, Sungchul Jung, Kihyun Hong, and Young-Yik Rhim. Ensemble based legal information retrieval and entailment system. In *The 10th International Workshop on Juris-Informatics (JURISIN)*, 2016.

[6] Mi-Young Kim and Ken Goebel, Randy Satoh. Coliee-2015 : Evaluation of legal question answering. In *Ninth International Workshop on Juris-informatics (JURISIN)*, 2015.

[7] Mi-Young Kim, Ying Xu, and Randy Goebel. Legal question answering using ranking svm and syntactic/semantic similarity. In *JSAI International Symposium on Artificial Intelligence*, pages 244–258. Springer, 2014.

[8] Mi-Young Kim, Ying Xu, Randy Goebel, and Ken Satoh. Answering yes/no questions in legal bar exams. In *JSAI International Symposium on Artificial Intelligence*, pages 199–213. Springer, 2013.

[9] Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. Neural architectures for named entity recognition. *arXiv preprint arXiv:1603.01360*, 2016.

[10] T Mikolov and J Dean. Distributed representations of words and phrases and their compositionality. *Advances in neural information processing systems*, 2013.

[11] George A Miller. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41, 1995.

[12] Truong-Son Nguyen, Viet-Anh Phan, Thanh-Huy Nguyen, Hai-Long Trieu, Ngoc-Phuong Chau, Trung-Tin Pham, and Le-Minh Nguyen. Legal information extraction/entailment using svm-ranking and tree-based convolutional neural network. In *The 10th International Workshop on Juris-Informatics (JURISIN)*, 2016.

[13] Ankur P Parikh, Oscar Täckström, Dipanjan Das, and Jakob Uszkoreit. A decomposable attention model for natural language inference. *arXiv preprint arXiv:1606.01933*, 2016.

[14] S. Pal Sushmita, A. Kanapala. Legal information retrieval task: Participation from ism, dhanbad. In *Ninth International Workshop on Juris-informatics (JURISIN)*, 2015.

[15] Satoshi Tojo. Eighth international workshop on juris-informatics (jurisin 2014). In *JSAI International Symposium on Artificial Intelligence*. Springer, 2014.

[16] Duc-Vu Tran, Viet-Anh Phan, Hai-Long Trieu, and Le-Minh Nguyen. An approach for retrieving legal texts. In *Ninth International Workshop on Juris-informatics (JURISIN)*, 2015.

[17] Oanh Thi Tran, Bach Xuan Ngo, Minh Le Nguyen, and Akira Shimazu. Answering legal questions by mining reference information. In *JSAI International Symposium on Artificial Intelligence*, pages 214–229. Springer, 2013.