Applying a Convolutional Neural Network to Legal Question Answering[1]

Mi-Young Kim, Ying Xu, and Randy Goebel

Alberta Innovates Centre for Machine Learning,
Department of Computing Science,
University of Alberta
Edmonton, Canada

{miyoung2,yx2,rgoebel}@ualberta.ca

Abstract. Our legal question answering system combines legal information retrieval and textual entailment, and we describe a legal question answering system that exploits a deep convolutional neural network. We have evaluated our system using the training/test data from the competition on legal information extraction/entailment (COLIEE). The competition focuses on the legal information processing related to answering yes/no questions from Japanese legal bar exams, and it consists of three phases: ad-hoc legal information retrieval, textual entailment, and a learning model-driven combination of the two phases. Phase 1 requires the identification of Japan civil law articles relevant to a legal bar exam query. For that phase, we have implemented a combined TF-IDF and Ranking SVM information retrieval component. Phase 2 requires the system to answer "Yes" or "No" to previously unseen queries, by comparing extracted meanings of queries with relevant articles. Our training of an entailment model focuses on features based on word embeddings, syntactic similarities and identification of negation/antonym relations. We augment our textual entailment component with a convolutional neural network with dropout regularization and Rectified Linear Units. To our knowledge, our study is the first to adapt deep learning for textual entailment. Experimental evaluation demonstrates the effectiveness of the convolutional neural network and dropout regularization. The results show that our deep learning-based method outperforms our baseline SVM-based supervised model and K-means clustering.

Keywords: legal question answering, recognizing textual entailment, information retrieval, convolutional neural network

1. Task Description

Legal question answering can be considered as a number of intermediate steps. For instance, consider a question such as "Is it true that a special provision that releases warranty can be made, but in that situation, when there are rights that the seller

[1] You can access the final publication version through https://doi.org/10.1007/978-3-319-50953-2_20

establishes on his/her own for a third party, the seller is not released of warranty?" In this example, a system must first identify and retrieve relevant documents, typically legal statutes, and subsequently, identify a most relevant sentence. Finally, it must compare the semantic connections between question and the relevant sentence, and determine whether an entailment relation holds.

Deep Neural Networks (DNNs) are an emerging technology that has recently demonstrated dramatic success in several areas, including speech feature extraction and recognition. Incorporation of convolution and subsequent pooling into a neural network has provided the basis for a technique called Convolutional Neural Networks (CNNs) [22]. CNNs have shown good performance in image and speech recognition [18], and many studies have proposed applying CNNs to natural language processing [11,12]. Here we adapt a CNN for legal question answering, especially focused on textual entailment. One primary motivation for using deeper models such as neural networks with many layers is that they have the potential to be much more representationally efficient compared with shallower neural network models. In textual entailment, we will extract linguistic features between two sentences, and determine textual entailment by comparing the features. In this task, not all linguistic features are directly related to each other, so we intend to capture related features, then connect them locally. One major motivation for CNNs is to restrict the network architecture through the use of local connections known as receptive fields.

The Competition on Legal Information Extraction/Entailment (COLIEE) 2015 focuses on two aspects of legal information processing related to answering yes/no questions from legal bar exams: legal document retrieval (Phase 1), and textual entailment for Yes/No question answering of legal queries (Phase 2).

Phase 1 is an ad-hoc information retrieval (IR) task. The goal is to retrieve relevant Japan civil law articles that are most relevant to a legal bar exam. We approach this problem with two models based on statistical information. One is the TF-IDF model [1], i.e., term frequency-inverse document frequency. The relevance between a query and a document depends on their intersection word set. The importance of words is measured with a function of term frequency and document frequency as parameters. Our terms are lemmatized words, e.g., the verbs "attending," "attends," and "attended" are lemmatized as the same base form "attend."

Another popular model for text retrieval is a Ranking SVM model [2]. That model is used to re-rank documents that are retrieved by the TF-IDF model. The features used to train this model are lexical words, dependency path bigrams and TF-IDF scores. The intuition is that the supervised model can learn weights or priority of words based on training data, in addition to or as an alternative to TF-IDF.

The goal of Phase 2 is to construct Yes/No question answering systems for legal queries, by confirming the entailment of questions from the relevant articles. The answer to a question is typically determined by measuring some kind of heuristically-determined semantic similarity between question and answer. While there are many possible approaches, we note that neural network-based distributional sentence models have achieved success in many natural language processing tasks such as sentiment analysis [12], paraphrase detection [13], document classification [14], and

question answering [11]. As a consequence of this success, it appears natural to approach textual entailment using similar techniques. Here we show that a neural network-based sentence model can be applied to the task of textual entailment. After constructing a set of pre-trained semantic word embeddings using the *word2vec* [20], we used a supervised method to learn a heuristic semantic-matching model between question and corresponding articles.

In addition to semantic word embeddings, our system uses features that depend on some components of the syntactic structure, and the presence of negation. We employ a convolutional neural network algorithm with dropout regularization and Rectified Linear Units, and compare its performance with baseline system based on support vector machines.

2. Phase 1: Legal Information Retrieval

2.1 IR Models

2.1.1 TF-IDF model

Here we introduce our TF-IDF and Ranking SVM models. Queries and articles are all tokenized and parsed by the Stanford NLP tool. For the IR task, the similarity of a query and an article is based on the terms within them. Our terms for TF-IDF are lemmatized words.

For TF-IDF, we use the simplified version of Lucene's similarity score of an article to a query as suggested in [15]:

$$tf\text{-}idf(Q,A) = \sum_{t}[\sqrt{tf(t,A)}\times\{1+\log(idf(t))\}^2]$$

The score *tf-idf(Q,A)* is a measure which estimates the relevance between a query *Q* and an article *A*. First, for every term *t* in the query *A*, we compute *tf(t,A),* and *idf(t)*. The score *tf(t,A)* is the term frequency of *t* in the article *A*, and *idf(t)* is the inverse document frequency of the term *t*, which is the number of articles that contain *t*. After some normalization computed within the Lucene package, we multiply *tf(t,A)* and *idf(t),* and then we compute the sum of these multiplication scores for all terms *t* in the query *A*. This summation result is *tf-idf(Q,A)*. The bigger *tf-idf(Q,A)* is, the more relevant between the query *Q* and the article *A*. The real version has some normalized parameters in terms of an article's length to alleviate the functions biased towards long documents. The parameters are set as the default of the Lucene's TF-IDF model.

2.1.2 Ranking SVM

The Ranking SVM model was proposed by Joachims [2]. That model ranks a set of retrieved documents based on a selection of attributes from user's data. Given the feature vector of a training instance, i.e., a retrieved article set given a query, denoted by *Φ(Q, Ai)*, the model tries to find a ranking that satisfies constraints:

$$\emptyset(Q,\ A_i) > \emptyset(Q,\ A_j)$$

where A_i is a relevant article for the query Q, while A_j is less relevant.

We adopt the same model and features suggested in [15]. The three types of features are as follows:

- Lexical words: the lemmatized normal form of surface structure of words in both the retrieved article and the query. In the conversion to the SVM's instance representation, this feature is converted into binary features whose values are one or zero, i.e., depending on if a word exists in the intersection word set or not.
- Dependency pairs: word pairs that are linked by a dependency link. The intuition is that, compared with the bag of words information, syntactic information should improve the capture of salient semantic content. Dependency parse features have been used in many NLP tasks, and improved IR performance [3]. This feature type is also converted into binary values.
- TF-IDF score (Section 2.1.1).

We set the Ranking SVM model's parameter c according to the cross validation on the training set.

2.2 Experiments

The legal IR task that we use to test our system has several sets of queries paired with the Japan civil law articles as documents (724 articles in total). Here follows one example of the query and a corresponding relevant article.

> *Question: A person who made a manifestation of intention which was induced by duress emanated from a third party may rescind such manifestation of intention on the basis of duress, only if the other party knew or was negligent of such fact.*
>
> *Related Article: (Fraud or Duress) Article 96 (1)Manifestation of intention which is induced by any fraud or duress may be rescinded. (2)In cases any third party commits any fraud inducing any person to make a manifestation of intention to the other party, such manifestation of intention may be rescinded only if the other party knew such fact.(3)The rescission of the manifestation of intention induced by the fraud pursuant to the provision of the preceding two paragraphs may not be asserted against a third party without knowledge.*

Before the final test set was released, we received 6 sets of queries for a "dry run" in COLIEE 2015. The 6 sets of data include 267 queries, and 326 relevant articles (average 1.22 articles per query). We used a corresponding 6-fold leave-one-out cross validation evaluation. The metrics for measuring our IR model performance is Mean Average Precision (MAP):

$$MAP(Q) = \frac{1}{|Q|}\sum_{q\in Q}\frac{1}{m}\sum_{k\in(1,m)} precision(R_k)$$

where Q is the set of queries, and m is the number of retrieved articles. R_k is the set of ranked retrieval results from the top until the k-th article. In the following experiments, we set m as 5 for all queries, corresponding to the column *MAP@5* in Table 1.

Table 1 shows the results of experiments with our two IR models on the legal IR task on the training set. The ensemble SVM-Ranking model is slightly better than the TF-IDF model. Table 2 shows the results of our SVM-ranking model on the final test set. The test data size is 79 queries for Phase 1. The performance of our system was ranked first among the submitted systems in the Competition on Legal Information Extraction/Entailment (COLIEE) 2015 [23].

Table 1. IR results on dry run data with different models.

Id	Models	MAP@5
1	TF-IDF with lemma	0.294
2	SVM-ranking	0.302

Table 2. IR results on test data using the SVM-ranking model

Participant ID	Performance on Phase 1	
UA (University of Alberta)	* The number of submitted articles: 79 * The number of correctly submitted articles: 50	
	Precision	0.6329
	Recall	0.4902
	F-measure	**0.5525**

3. Phase 2: Answering 'Yes'/'No' Questions Using a Convolutional Neural Network

Our system uses syntactic information in addition to word embedding to predict textual entailment. We exploit syntactic similarity features, negation and antonyms in Kim et al. [15]. Details are provided in the next subsections.

3.1 Our System

3.1.1 Model description

The problem of answering a legal yes/no question can be viewed as a binary classification problem. Assume a set of questions Q, where each question $q_i \in Q$ is associated with a list of corresponding article sentences $\{a_{i1}, a_{i2}, ..., a_{im}\}$, where $y_i = 1$ if the answer is 'yes' and $y_i = 0$ otherwise. We choose the most relevant sentence a_{ij}

using the algorithm of Kim et al. [15], and we simply treat each data point as a triple (q_i, a_{ij}, y_i). Therefore, our task is to learn a classifier over these triples so that it can predict the answers of any additional question-article pairs.

Our solution assumes that correct answers have high semantic similarity to questions. We model questions and answers as vectors using word embedding and linguistic information, and evaluate the relatedness of each question-article pair in a shared vector space. Following Yu et al. [11], given the vector representations of a question $\boldsymbol{q}$ and a most relevant article sentence $\boldsymbol{a}$, the probability of the answer being correct is

$$p(y = 1|\boldsymbol{q}, \boldsymbol{a}) = rectifier(\boldsymbol{q}^T \boldsymbol{M} \boldsymbol{a} + \boldsymbol{b})$$

Where the bias term b and the transformation matrix $\boldsymbol{M}$ in $R^{d\times d}$ are model parameters. This formulation can be understood as follows: we first generate a question through the transformation $\boldsymbol{q'}=\boldsymbol{Ma}$, and then measure the similarity of the generated question $\boldsymbol{q'}$ and the given question $\boldsymbol{q}$ by their dot product. The rectifier function is used as an activation function.

As mentioned above, a Convolutional Neural Network(CNN) is a biologically-inspired variant of a multi-layer perceptron. CNN employs two component techniques: (1) restricting the network architecture through the use of local connections known as receptive fields; and (2) constraining the choice of synaptic weights through the use of weight-sharing. Most of the applications of CNNs include a max-pooling layer, which reduces and integrates the neighboring neurons' outputs. CNNs also exploit spatially-local correlation by enforcing a local connectivity pattern between neurons of adjacent layers.

CNN-based models have been proved to be effective in applications such as twitter sentiment prediction [12] and semantic parsing [16]. Figure 1 illustrates the architecture of the CNN-based sentence model in one dimension. We use word embedding and linguistic features with one convolutional layer and one pooling layer. A word embedding is a parameterized function mapping words in some language to high-dimensional vectors. We use the Bag-of-words model of [11] for word embedding.

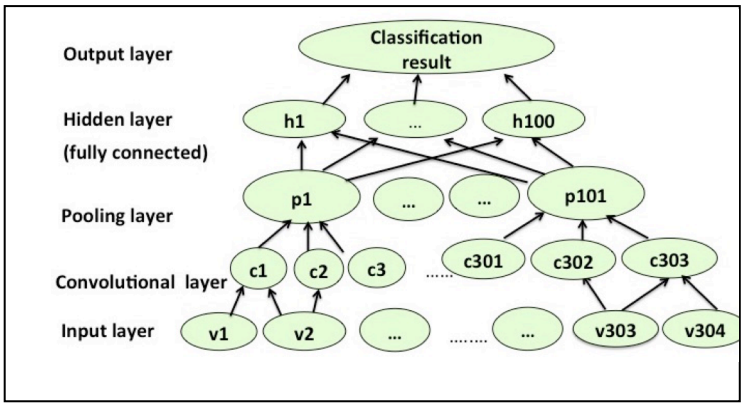

Figure 1. Our CNN architecture

Given word embeddings, the bag-of-words model generates the vector representation of a sentence by summing over the embeddings of all words in the sentence, after removing stopwords. The vector is then normalized by the length of the sentence.

$$s = \frac{1}{n}\sum_{i=1}^{n} s_i$$

In the formula above, s and s_i are d-dimensional vectors. s_i corresponds to the vector of the *i*-th word in the sentence, *n* is the number of words in the sentence, and *s* is vector representation of the sentence. We used *word2vec* [20] word embedding technique (d=50, which is typically used), and used the training data of COLIEE 2014 for training *word2vec*. *Word2vec* [20] used a neural network consisting of input layer, projection layer, and output layer and they removed a hidden layer to improve learning speed.

In addition to word embeddings, the types of features we use are as follows:

a) Word Lemma
b) Tree structure features (considering only roots)

Feature 1 : $w_{root}(condition_{query\_n})$
Feature 2: $w_{root}(condition_{article\_n})$
Feature 3: $w_{root}(conclusion_{query\_n})$
Feature 4: $w_{root}(conclusion_{article\_n})$
Feature 5: neg_level($condition_{query\_n}$)
Feature 6: neg_level($condition_{article\_n}$)
Feature 7: neg_level($conclusion_{query\_n}$)
Feature 8: neg_level($conclusion_{article\_n}$)

In Figure 1, the input layer consists of the following word embedding vectors and binary values:

1) v1, v3,..., v99 (odd index of nodes between v1 and v99): word embedding vector of the query sentence
2) v2, v4, ..., v100 (even index of nodes between v2 and v100): word embedding vector of the relevant article sentence
3) v101, v103, ..., v199 (odd index of nodes between v101 and v199): word embedding vector of the Feature 1
4) v102, v104, ..., v200 (even index of nodes between v102 and v200): word embedding vector of the Feature 2
5) v201, v203, ..., v299 (odd index of nodes between v201 and v299): word embedding vector of the Feature 3
6) v202, v204, ..., v300 (even index of nodes between v202 and v300): word embedding vector of the Feature 4
7) v301-v304 : binary values of the Features 5-8

In the features above, $article_n$ is the most relevant article of the query $query_n$. First we detect condition part and conclusion part in the question and corresponding article,

and also compute negation value (neg_level()) of each part according to Kim et al. [15]. The following is an example of condition and conclusion detection:

<Civil Law Article 177> *Acquisitions of, losses of and changes in real rights concerning immovable properties may not be asserted against third parties, unless the same are registered pursuant to the applicable provisions of the Real Estate Registration Act (Law No. 123 of 2004) and other laws regarding registration.*

(1) Conclusion => Acquisitions of, losses of and changes in real rights concerning immovable properties may ***not*** *be asserted against third parties,*
(2) Condition => ***unless*** *the same are registered pursuant to the applicable provisions of the Real Estate Registration Act (Law No. 123 of 2004) and other laws regarding registration.*

In Features 1-4, $w_{root}(s)$ means the root word in the syntactic tree of the sentence *s*. Features 1-4 consider both lexical and syntactic information, and Features 5-8 incorporate negation and antonym information. We use some morphological and syntactic analysis to extract lemma and dependency information. Details of the morphological and syntactic analyzer are given in Section 3.2.

As shown in Figure 1, the nodes of the input layer of even indices between v1 and v100 (such as v2, v4, v6.... and v100) indicate the word embedding vector for a relevant article sentence, and the nodes of odd indices between v1 and v100 (such as v1, v3, ... and v99) show the word embedding vector for a query sentence. The nodes between v101 and v304 are linguistic features. Because the adjacent two nodes indicate the same feature type (e.g., v1 and v2 indicate the first index value of each word embedding vector), we make a convolutional layer constructed from the adjusting two input nodes.

The convolutional vector $\boldsymbol{t}$ in R^2 (the 2-dimensional real number space) projects adjacent two nodes into a feature value c_i, computed as follows:

$$C_i = rectifier(\boldsymbol{t} * v_{i:i+1} + b),$$

where *rectifier(x) = max(0,x)*. We explain the rectifier function in the subsection 3.1.2.

In the pooling layer, we just do summation of 3 adjacent c_i values, to reduce the features. The number 3 was just chosen for this experiment, and we can find an optimal number in future work. The p_i values in the pooling layer as follows:

$$p_i = \sum_{k=3i-2}^{3i} c_i \qquad i = 1,2,\dots,101$$

The training is done with multithreaded mini-batch gradient descent in the weka[2] tool.

3.1.2 Dropout regularization and Rectified Linear Units

When a neural network is trained on a small training set, it typically performs poorly on test data. This "overfitting" is greatly reduced by randomly omitting some of the feature detectors on each training case. It is called 'dropout'. The dropout prevents complex co-adaptations in which a feature detector is only helpful in the context of several other specific feature detectors. Random dropout gives big improvements on many benchmark tasks and sets new records for speech and object recognition [21]. We found that the dropout rate needs to be between 0.6 and 0.7 for a hidden layer and 0.1 for an input layer in order to make it effective in achieving low errors.
We also employ the Rectified Linear Unit for CNN. Neural networks with rectified linear unit (ReLU) non-linearities have been highly successful for computer vision tasks and have been shown to be faster to train than standard sigmoid units [17].

ReLU is a neuron which uses a rectifier instead of a hyperbolic tangent or logistic function as an activation function. *Rectifier f(x)=max(0,x)* allows a network to easily obtain sparse representations.

3.1.3 Supervised learning with SVM

We have compared our method with SVM, as a kind of supervised learning model. Using the SVM tool included in the Weka [4] software, we performed cross-validation for the 179 questions of the dry run data in COLIEE 2014 using word embedding vector and linguistic features in Section 3.1.1. We used a linear kernel SVM because it is popular for real-time applications as they enjoy both faster training and classification speeds, with significantly reduced memory requirements than non-linear kernels, because of the compact representation of the decision function.

3.2 Experimental setup for Phase 2

In the general formulation of the textual entailment problem, given an input text sentence and a hypothesis sentence, the task is to make predictions about whether or not the hypothesis is entailed by the input sentence. We report the accuracy of our method in answering yes/no questions of legal bar exams by predicting whether questions are entailed by the corresponding civil law articles.

There is a balanced positive-negative sample distribution in the dataset (55.87% yes, and 44.13% no) for a dry run of COLIEE 2014 dataset, so we consider the baseline for true/false evaluation is the accuracy when returning always “yes,” which is 55.87%. Our data for our dry run has 179 questions.

The original examinations are provided in Japanese and English, and our initial implementation used a Korean translation, provided by the Excite translation tool

[2] https://weka.wikispaces.com/Unofficial+packages+for+WEKA+3.7

(http://excite.translation.jp/world/). The reason that we chose Korean is that we have a team member whose native language is Korean, and the characteristics of Korean and Japanese language are similar. In addition, the translation quality between two languages ensures relatively stable performance. Because our study team includes a Korean researcher, we can easily analyze the errors and intermediate rules in Korean. We used a Korean morphological analyzer and dependency parser [5].

3.3 Experimental results

To compare our performance with Kim et al. [15], we measured our system's performance on the dry run data of COLIEE 2014. Table 3 shows the experimental results. An SVM-based model showed accuracy of 60.12%, and a convolutional neural network with pre-trained semantic word embeddings and dropout showed best performance of 63.87% with the setting of input layer dropout rate of 0.1, hidden layer dropout rate of 0.6, and 100 hidden layer nodes. When we did not use the dropout regularization, the accuracy was lower by 1.22%. Without dropout and word embedding, the accuracy was 56.30%, which showed no significant difference with the baseline accuracy. We also compare our performance with the Kim et al.'s [15] performance using the same dataset. In [15], they used their linguistic features for SVM learning, and also proposed a model combining rule-based method and k-means clustering. Our CNN performance outperformed both of their SVM model and combined model.

Table 3. Experimental results on dry run data for Phase 2

Our method	Accuracy (%)
Baseline	55.87
Cross-validation with Supervised learning (SVM) [15]	59.43
Rule-based model + K-means clustering [15]	61.96
Cross-validation with Supervised learning (SVM) using our features	60.12
Convolutional Neural Network with Word Embedding + Linguistic features + Dropout	63.87
Convolutional Neural Network with Word Embedding +Linguistic features	62.65
Convolutional Neural Network with only linguistic features	56.30

Table 4. Experimental Results on the formal run data of COLIEE 2015

Our method	Accuracy(%)
Entailment results (Phase 2)	66.67
Combined results (Phase 3)	65.82

Table 5. Error types

Error type	Accuracy (%)	Error type	Accuracy(%)
Specific example case	6.28	Paraphrasing	38.85
Exceptional case	9.14	Complex constraints in condition	28.09
Incorrect detection of condition, conclusion	3.84	Reference to another article	4.10
Etc.	9.7		

Table 4 shows the experimental results using formal run data of COLIEE 2015. The formal run data size of COLIEE 2015 is 66 queries for Phase 2 from the bar exam of 2013. For Phase 3, we use the same test data of Phase 1, which consists of 79 queries extracted from the bar exam of 2012. Our performance of textual entailment (phase 2) is 66.67%, and the performance of combined phase (phase 3) is 65.82%. This result is ranked first among the systems in COLIEE 2015 competition [23].

From unsuccessful instances, we classified the error types as shown in Table 5. We could not identify the errors arising from the Neural Network architecture or embedding vectors, so we just classified the errors into 7 cases which are shown in Table 5. The biggest error arises, of course, from the paraphrasing problem. The second biggest error is because of complex constraints in conditions. As with the other error types, there are cases where a question is an example case of the corresponding article, and the corresponding article embeds another article. There has been also errors in the case that a question is an exceptional case of the corresponding legal law article. In further work, we will need to complement our knowledge base with some kind of paraphrasing dictionary employing a paraphrase detection method.

4 Related work

Only very recently have researchers started to apply deep learning to question answering [11,16,19]. Relevant work includes Yih et al. [16] who constructed models for single-relation question answering with a knowledge base of triples. In the same direction, Bordes et al. [19] used a type of siamese network for learning to project question and answer pairs into a joint space. Finally, Yu et al. [11] selected answer sentences, which includes the answer of a question. They modelled semantic composition with a recursive neural network. However these tasks differ from the

work presented here in that our purpose is not to make a choice of answer selection in a document, but to answer “yes” or “no.”

A textual entailment method from W. Bdour et al. [6] provided the basis for a Yes/No Arabic Question Answering System. They used a kind of logical representation, which bridges the distinct representations of the functional structure obtained for questions and passages. This method is also not appropriate for our task. If a false question sentence is constructed by replacing named entities with terms of different meaning in the legal article, a logic representation can be helpful. However, false questions are not simply constructed by substituting specific named entities, and any logical representation can make the problem more complex. Nielsen et al. [7] extracted features from dependency paths, and combined them with word-alignment features in a mixture of an expert-based classifier. Zanzotto et al. [8] proposed a syntactic cross-pair similarity measure for RTE. Harmeling [9] took a similar classification-based approach with transformation sequence features. Marsi et al. [10] described a system using dependency-based paraphrasing techniques. All these previous systems uniformly conclude that syntactic information is helpful in RTE: we also use syntactic information.

As further research, we will try unsupervised pre-training with a CNN to solve the problem of small training datasets. We are also considering adopting more convolutional layers and pooling layers in the CNN architecture, and investigating the effect of more layers in the textual entailment problem. The challenge is the management of the tradeoff between encoding attribute dependencies and learning effective models.

5 Conclusion

We have described our implementation for the Competition on Legal Information Extraction/Entailment (COLIEE) Task.

For Phase 1, legal information retrieval, we implemented a Ranking-SVM model for the legal information retrieval task. By incorporating features such as lexical words, dependency links, and TF-IDF score, our model shows better mean average precision than TF-IDF.

For Phase 2, we have proposed a method to answer yes/no questions from legal bar exams related to civil law. We used a convolutional neural network model using dropout regularization and Rectified Linear Units with pre-trained semantic word embeddings. We also extract deep linguistic features with lexical, syntactic information based on morphological analysis and dependency trees. We show the improved performance over previous systems, using a convolutional neural network.

Acknowledgements

This research was supported by the Alberta Innovates Centre for Machine Learning (AICML) and the Natural Sciences and Engineering Research Council (NSERC). We

are indebted to Ken Satoh of the National Institute for Informatics, who has had the vision to create the COLIEE competition.

References

1. K. Sparck Jones, A statistical interpretation of term specificity and its application in retrieval. In: Willett, P. (ed.) Document Retrieval Systems, pp. 132-142. Taylor Graham Publishing, London, UK, UK, 1988
2. T. Joachims, Optimizing search engines using clickthrough data. In: Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. pp. 133-142. KDD '02, ACM, New York, NY, USA, 2002
3. K.T. Maxwell, J. Oberlander, W.B. Croft. Feature-based selection of dependency paths in ad hoc information retrieval. In: Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). pp. 507-516. Association for Computational Linguistics, Sofia, Bulgaria, August 2013
4. M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, I. H. Witten, The WEKA Data Mining Software: An Update; SIGKDD Explorations, Volume 11, Issue 1. 2009
5. M-Y. Kim, S-J. Kang and J-H. Lee, Resolving Ambiguity in Inter-chunk Dependency Parsing, Proc. of 6th Natural Language Processing Pacific Rim Symposium, pp. 263-270, 2001
6. W. N. Bdour, and N.K. Gharaibeh, Development of Yes/No Arabic Question Answering System, International Journal of Artificial Intelligence and Applications, Vol.4, No.1 (51-63), 2013
7. R. D. Nielsen, W. Ward, and J. H. Martin. Toward dependency path based entailment. In Proceedings of the second PASCAL Challenges Workshop on RTE, 2006
8. F. M. Zanzotto, A. Moschitti, M. Pennacchiotti, and M.T. Pazienza. Learning textual entailment from examples. In Proceedings of the second PASCAL Challenges Workshop on RTE, 2006
9. S. Harmeling, An extensible probabilistic transformation-based approach to the third recognizing textual entailment challenge. In Proceedings of ACL PASCAL Workshop on Textual Entailment and Paraphrasing, 2007
10. E. Marsi, E. Krahmer, and W. Bosma. Dependency-based paraphrasing for recognizing textual entailment. In Proceedings of ACL PASCAL Workshop on Textual Entailment and Paraphrasing, 2007
11. L. Yu, K. M. Hermann, P. Blunsom, and S. Pulman. Deep learning for answer sentence selection. arXiv preprint arXiv:1412.1632. 2014
12. N. Kalchbrenner, E. Grefenstette, and P. Blunsom. A convolutional neural network for modelling sentences. In Proceedings of ACL, 2014.
13. R. Socher, B. Huval, C. D. Manning, and A. Y. Ng. Semantic compositionality through recursive matrix-vector spaces. In Proceedings of EMNLP-CoNLL, 2012.
14. K. M. Hermann and P. Blunsom. Multilingual Models for Compositional Distributional Semantics. In Proceedings of ACL, 2014.
15. M-Y. Kim, Y. Xu, and R. Goebel. Alberta-KXG: Legal Question Answering Using Ranking SVM and Syntactic/Semantic Similarity. JURISIN Workshop, 2014
16. Wen-tau Yih, Xiaodong He, and Christopher Meek. Semantic parsing for single-relation question answering. In Proceedings of ACL, 2014.
17. G. E. Dahl, T. N. Sainath, and G. E. Hinton, Improving deep neural networks for LVCSR using rectified linear units and dropout. In Proceedings of Acoustics, Speech and Signal Processing (ICASSP), pp. 8609-8613, 2013

18. L. Deng, O. Abdel-Hamid, and D. Yu. "A deep convolutional neural network using heterogeneous pooling for trading acoustic invariance with phonetic confusion." In Proceedings of Acoustics, Speech and Signal Processing (ICASSP), pp. 6669-6673. IEEE, 2013.
19. Antoine Bordes, Sumit Chopra, and Jason Weston. Question answering with subgraph embeddings. In Proceedings of EMNLP, 2014.
20. T. Mikolov, I. Sutskever, K. Chen, G.S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In Advances in neural information processing systems pp. 3111-3119, 2013
21. G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R.R. Salakhutdinov, Improving neural networks by preventing co-adaptation of feature detectors. arXiv preprint arXiv:1207.0580, 2012
22. M. Kouylekov, and B. Magnini. Tree edit distance for recognizing textual entailment: Estimating the cost of insertion. In Proceedings of the second PASCAL Challenges Workshop on RTE, 2006
23. M. Kim, R. Goebel and K. Satoh. COLIEE-2015: Evaluation of Legal Question Answering. In Ninth Workshop on Juris-informatics (JURISIN), 2015