EPiC Series in Computing

Volume 47, 2017, Pages 68–78

COLIEE 2017. Fourth Competition on Legal Information Extraction/Entailment

Legal Information Retrieval Using Topic Clustering and Neural Networks

Rohan Nanda[1], Adebayo Kolawole John[1], Luigi Di Caro[1], Guido Boella[1], and Livio Robaldo[2]

[1] University of Turin, Italy
nanda@di.unito.it, kolawolejohn.adebayo@unibo.it, {dicaro,guido}@di.unito.it
[2] University of Luxembourg
livio.robaldo@uni.lu

Abstract

This paper presents a description about our adopted approach for the information retrieval and textual entailment tasks of the COLIEE 2017 competition. We address the information retrieval task by implementing a partial string matching and a topic clustering method. For the textual entailment task, we propose a Long Short-Term Memory (LSTM) - Convolutional Neural Network (CNN) model which utilizes word embeddings trained on the Google News vectors. We evaluated our approach for both tasks on the COLIEE 2017 dataset. The results demonstrate that the topic clustering method outperformed the partial string matching method in the information retrieval task. The performance of LSTM-CNN model was competitive with other textual entailment systems.

1 Introduction

The COLIEE (Competition on Legal Information Extraction/Entailment) 2017 comprises two tasks : information retrieval and textual entailment. The information retrieval task deals with identifying a set of Civil Code articles for a given question or query. The information retrieval system takes a query Q as input and retrieves a set of articles (from the entire Civil Code) relevant to the query. The textual entailment task deals with recognizing textual entailment between the query and the retrieved articles. The objective is to determine if the article confirms "yes" or "no" as an answer to the query.

In this paper[1], we address both tasks of information retrieval and textual entailment. In the next section, we discuss our approach to the information retrieval task. Section 3 describes our approach for the textual entailment task. Section 4 presents the results and analysis. The paper concludes in Section 5.

2 Task 1: Information Retrieval

The objective of information retrieval task is to retrieve a set of articles which are relevant for the input query. Text similarity techniques are an integral part of information retrieval system. Lexical

[1]The first and second author contributed equally

similarity techniques use string-based algorithms for measuring the similarity between two text strings [9]. However, they don't take into account the ambiguities of natural language. Semantic similarity methods comprise knowledge-based and corpus-based techniques. Corpus-based techniques measure the degree of similarity between texts by learning information from a large corpora. Latent semantic analysis (LSA) is one such method which is based on the assumption that semantically similar words occur in similar pieces of text [19]. Knowledge-based techniques compute the degree of similarity between texts by utilizing the knowledge from semantic networks. Topic-based models assign topics to each document where each topic is represented by a term distribution [1]. Semantically similar documents are assigned similar topics. Recently, new corpus-based techniques like word embeddings have been utilized to compute semantic similarity for information retrieval systems [16]. They utilize a neural network language model to predict a word from its surrounding words (continuous bag-of-words model) or predict several surrounding words from an input word (Skip-gram model). These models have shown promising results when trained on a large corpus. However, the COLIEE corpus is quite small as it contains short text articles instead of documents. The total number of articles in the corpus are around 1000. Therefore we tailor our approach for the COLIEE dataset.

The manual analysis of the corpus suggested that in many cases, the length of queries was much shorter than the length of the corresponding articles. The query in some cases only partially expressed the semantics of the article. Also the query and corresponding article had few similar words or phrases in common. Therefore, as our first approach for Task 1 we decided to use a partial string matching algorithm. In this approach, each query is compared with all the articles in the corpus to retrieve the most similar article. The texts of the query Q and article A are first tokenized. Each group of tokens in Q and A is considered as a set [24]. Then the intersection set, I of sorted tokens in set Q and set A is derived as follows:

$$I = Q \cap A \tag{1}$$

The Query set Q is represented as the union of the tokens in the intersection set I and the remaining tokens in the remainder query set R_Q.

$$Q = I \cup R_Q \tag{2}$$

The article set A can be represented as the union of the intersection set I and the remainder article set R_A.

$$A = I \cup R_A \tag{3}$$

Further, we compute three similarity measures: (I,Q), (I,A) and (Q,A). The similarity measure PS between two sets is computed as $2.0 * M/T$, where T is the total number of elements in both sets and M is the number of matches [24]. The maximum similarity value of the three is considered as the partial string similarity. The similarity is in the range of [0,1]. The major significance of this method is the intersection set I. The query set Q and article set A have a high similarity value when set I is the larger part of either set A or Q.

The first approach makes a naive assumption that *relevance* is a measure of the number of common tokens between an article and a query. This assumption is not totally valid, considering that synonyms and polysemous words appear frequently in natural language texts. Our second approach takes care of the naivety by tackling relevance as a measure of semantic similarity between an article and the query. Usually, instead of computing this similarity between each query and all the articles present, we group articles into clusters of topic such that each article has a topic vector which is compared to that of the query. For each query, we only focus on the articles with similar topic vectors. The top-n most similar articles to the query are then passed through the semantic similarity module. This overcomes the limitation of the first approach in two ways. First, by adopting an initial *topic-based* clustering solution, we overcome the bias of tokens that have less importance in the articles as well as language variability issues like synonymy and polysemy. Also, we limit the relevancy search to a subset of 'supposedly' relevant articles. We call this set of articles the *seed* set. Secondly, by computing the semantic similarity between the query and articles in its seed set, we limit the over-dependence on language units like words and their occurrence counts, while considering the interplay between the words in both the query and article.

2.1 Clustering With Topic Models

The first step in our second approach is to cluster articles into their respective topics, thus yielding a set of low dimensional topic vectors for each article. This can easily be done with topic modeling of some documents. Topic Models (TM) are a suite of algorithm that unearth the latent topic distribution in a set of documents, based on the assumption that a document is a mixture of topics. Popular topic models are the Latent Semantic analysis (LSA) [7] and the Latent Dirichlet Allocation (LDA) [5]. We used LDA for this part of our work. LDA is a generative probabilistic model with the intuition that a document is a random distribution over latent topics, where each topic is characterized by a distribution over words in the vocabulary. The model has shown capability to capture semantic information from documents in a way similar to probabilistic latent semantic analysis [11] such that a low dimensionality representation of texts is produced in the semantic space while preserving their latent statistical features.

Formally, given a document $\mathbf{w}$ of N words such that $\mathbf{w} = (\mathrm{w}_1,\mathrm{w}_2,\mathrm{w}_3,...\mathrm{w}_N)$ and a corpus D of M documents denoted by $D = (\mathbf{w}_1,\mathbf{w}_2,\mathbf{w}_3.....\mathbf{w}_M)$. For each of the words w_n in the document, a topic z_n is drawn from the topic distribution θ, and a word w_n is randomly chosen from $\mathrm{P}(\mathrm{w}_n \mid \mathrm{z}_n, \beta)$ conditioned on z_n. Given α, a k-vector with components with $\alpha_i > 0$ and the Gamma function $\Gamma(\mathrm{x})$. The probability density of the Dirichlet is given as

$$P(\Theta|\alpha) = \frac{\Gamma(\sum_{i=1}^{k} \alpha_i)}{\Pi_{i=1}^{k}\Gamma(\alpha_i)}\Theta_1^{\alpha_1-1}....\Theta_k^{\alpha_k-1} \tag{4}$$

Given the parameters α and β, the joint distribution of a topic mixture θ, a set of N topics $\mathbf{z}$, and a set of N words $\mathbf{w}$ is thus given by

$$P(\theta, \mathbf{z}, \mathbf{w}|\alpha, \beta) = P(\theta|\alpha)\Pi_{n=1}^{N}P(z_n|\theta)P(w_n|z_n, \beta) \tag{5}$$

Integrating over θ and summing of z, the set of topic assignments, the distribution of a document can be obtained as shown in equation (6) below

$$P(\mathbf{w}|\alpha, \beta) = \int P(\theta|\alpha) \left(\Pi_{n=1}^{N}\sum_{z_n} P(z_n|\theta)P(w_n|z_n, \beta)\right) \mathrm{d}\theta \tag{6}$$

where $\mathrm{P}(\mathrm{z}_n \mid \theta)$ is θ_i for the unique i such that $z_n^i = 1$ The probability of a corpus is obtained through the product of marginal probability above for each $\mathbf{w}_n$ in D as given in the equation (7) below:

$$P(\mathbf{w}|\alpha, \beta) = \{\Pi_{d=1}^{M} \int P(\theta_d|\alpha)\Big(\Pi_{n=1}^{N_d} \sum_{z_d n} P(z_d n|\theta_d)P(w_d n|z_d n, \beta)\Big) \mathrm{d}\theta_d\} \tag{7}$$

Training LDA model on a corpus requires feeding the model with sets of tokens from the document. The model statistically estimates the topic distribution θ_d for each document as well as the word distribution in each topic. A model can also be used to predict topic classes for a previously unseen document. More information about LDA can be found in [5, 4]. In order to train LDA algorithm, we used all the articles in the training set as well the queries, taking pair of article-query as a document. We used known relevant pairings of query-article where available and for every other case, we feed each of the query or article as independent document. Furthermore, we include extra articles which formed parts of the documents released for COLIEE 2017 training set. We use the Gensim[2] implementation of LDA with Gibbs sampling, with T = 25 topics and the number of inference also being 20. All the documents

[2]Gensim is a python library offering a suite of algorithms for text processing. Available at https://radimrehurek.com/gensim/

were stemmed and stopwords removed. Each document was weighted with term frequency-inverse document frequency (tfidf) values in order to overcome the bias against unimportant but frequent words. Even though the number of topics was intuitively chosen, we observed that setting the number of topics to be smaller makes the topic distribution to be dense such that most articles have similar topical distribution. Also, when we increased the topic number, we identified a lot of arbitrariness and imprecision in topic distribution. We assumed that this observation is probably due to the tiny size of the training data used.

We use the resulting LDA model to generate topic distribution for each training articles such that the set of topic IDs are transformed into a dense topic vector of 25-dimension. First, we obtain a matrix $G = L \times T$ where $l \in L$ is a sparse vector of length T, the chosen number of topics. Each vector l contains the distribution of the topic IDs assigned by LDA to each article, where by topic ID, we denote the topic group or cluster that an article belongs, i.e., a number in the range $[0,\ T-1]$. The procedure is also repeated for the queries. The resulting outputs are two matrices transformed to dense, i.e. article and query, all of the same dimension. We used Faiss[3] to index these matrices. Faiss implements an array of search algorithms for search and clustering of dense vectors of documents and queries. These algorithms include exact match, L2, dot product vector comparison, nearest neighour, k-means, cosine similarity etc.

For each query, we pick the top 5 matching articles. Now, these articles are not assumed to be the exact relevant ones but rather selected as the seed set and input to the semantic similarity module. The semantic similarity module computes a similarity score between the text of each of the documents in the seed set and the query. Specifically, this module takes account of semantic information and word order information in the text of both the query and the article. Furthermore, it uses WordNet as a lexical taxonomy for looking up synonyms of words while calculating similarity. The similarity function between two words *w1* and *w2* is defined as the product of the depth function, *f1(h)* and length function, *f2(l)* as follows [13]:

$$S(w1, w2) = f1(h).f2(l) \tag{8}$$

The length function *f2(l)* is a monotonically decreasing function of path length l. It includes a constant alpha.

$$f2(l) = e^{-\alpha.l} \tag{9}$$

The depth function is a monotonically increasing function of depth h in concept hierarchy. It is represented as follows:

$$f1(h) = \frac{e^{\beta.h} - e^{-\beta.h}}{e^{\beta.h} + e^{-\beta.h}} \tag{10}$$

The values of alpha and beta are set to 0.2 and 0.45 respectively as per Li et al. [20].

Once the similarity scores have been computed, we sort the articles according to the similarity scores from highest to lowest. We pick the article with the highest similarity score along with any other that has a distance of not more than 15% to the topmost similar article, i.e., the one with the highest similarity score from the semantic similarity module. We did not evaluate, how the choice of the chosen parameters (threshold and top-5) impacts the result and such analysis is planned for future work. The distance score, *Dist*, is computed according to the formula in equation (11).

$$Dist = \Big(\frac{Hsim - aSim}{Hsim}\Big) \times 100 \tag{11}$$

where *Hsim* is the similarity score (highest of the top-5) of the most similar articles according to the similarity module and *aSim* is a similarity score of any of the other articles in the top-5.

3 Task2: Textual Entailment

The second task of COLIEE 2017 involves determining whether the retrieved articles in the first task entail the corresponding queries or not. This task is referred to as Textual Entailment (TE) and its goal

[3]Faiss is available at https://github.com/facebookresearch/faiss.

is to identify entailment relationship between a text and an hypothesis [3]. A premise or text entails an hypothesis (which is another text) if the hypothesis can be inferred from the premise. TE is a typical classification task, either as a binary (YES/NO) or multi-class classification (YES/NO/NEUTRAL etc.). The organizers of COLIEE2017 proposed a binary classification task.

Early TE systems rely on the use of hand crafted features [8]. In addition, a lot of knowledge resources, e.g., Wordnet are often employed in developing these systems [22]. The features often extracted from the training text include simple ones like n-gram overlap, string matching, presence of negation words e.g., never, shouldn't etc. Also, information retrieval (IR)-inspired features such as the TFIDF and a measure of similarity between the text and the hypothesis obtained with metrics such as the cosine similarity etc, are often used [12].

Even though a number of the systems which rely on feature engineering have achieved relative success, the efforts required in engineering features and feature selection pales this success. An evaluation of systems using different techniques for solving TE tasks is detailed in [3].

Since Neural Networks (NNs) have continued to give excellent performance in language modeling tasks [15], researchers have employed some NNs models for TE tasks as well. Specifically, the listed papers [6, 23, 2] show that deep Neural architecture can match and outperform lexical classifiers which use hand-crafted features for TE task. While these systems are very promising, they enjoy the benefit of a reasonable large amount of training data. Therefore, as we will show later in the results section, the performance reported in this paper may not be competitive compared to these works, given the very tiny training set available in this particular COLIEE challenge.

The authors in [17] utilized a Convolutional Neural Networks (CNN) to model entailment relationship on a previous COLIEE competition dataset. Also, the work described in [14] which was also based on a past COLIEE competition employed the use of Long Short-Term Memory (LSTM) for modeling entailment relationship.

We utilize NNs to autonomously learn latent enatailment features from a pair of text and hypothesis. Our NNs model combines LSTM [10] with CNN [18]. LSTM Networks are a powerful type of recurrent neural networks (RNNs) and are robust to the vanishing gradient problem [10]. Also, LSTMs generally have enhanced memory since they are able to retain information over many time steps. CNNs on the other hand can capture multiple features between adjacent input, owing to the many convolution it uses to filter the input sequences. Each filter transforms a local patch of lower-level features into higher-level representation [18, 15]. The LSTM state transition is given in equation (12) below.

$$
\begin{aligned}
i_t &= \sigma\Big(W^{(i)}x_t + U^{(i)}h_{t-1} + b^{(i)}\Big),\\
f_t &= \sigma\Big(W^{(f)}x_t + U^{(f)}h_{t-1} + b^{(f)}\Big),\\
o_t &= \sigma\Big(W^{(o)}x_t + U^{(o)}h_{t-1} + b^{(o)}\Big),\\
u_t &= \tanh\Big(W^{(u)}x_t + U^{(u)}h_{t-1} + b^{(u)}\Big),\\
c_t &= i_t \odot u_t + f_t \odot c_{t-1},\\
h_t &= o_t \odot \tanh c_t
\end{aligned}
\tag{12}
$$

Figure 1 gives a pictorial illustration of the model used in this work. Our model is quite simple, and it is based on sentence encoding approach. This was intentional since we have a very small training set. Also, even when we tried more complex model architectures, we keep getting similar performances.

First, at every input point, the sequences of both the article and the query sentences are each embedded into a 300 dimensional vectors. Instead of learning embeddings from scratch, we used the *GoogleNews* vectors for this part of our work. The *GoogleNews* vectors is a result of training the word2vec algorithm [21] on over 3 billion words. Also, we keep the weights of the embeddings frozen throughout all the layers. Note that the encoded articles and queries have the same sequence length, as they have been zero-padded in order to equal the maximum sequence length (*774*) in the input set. For each data sample, the input sequences are tokens of the corresponding articles and queries. At each time step, the vectors of a word from an input sequence is obtained. We associate each token t

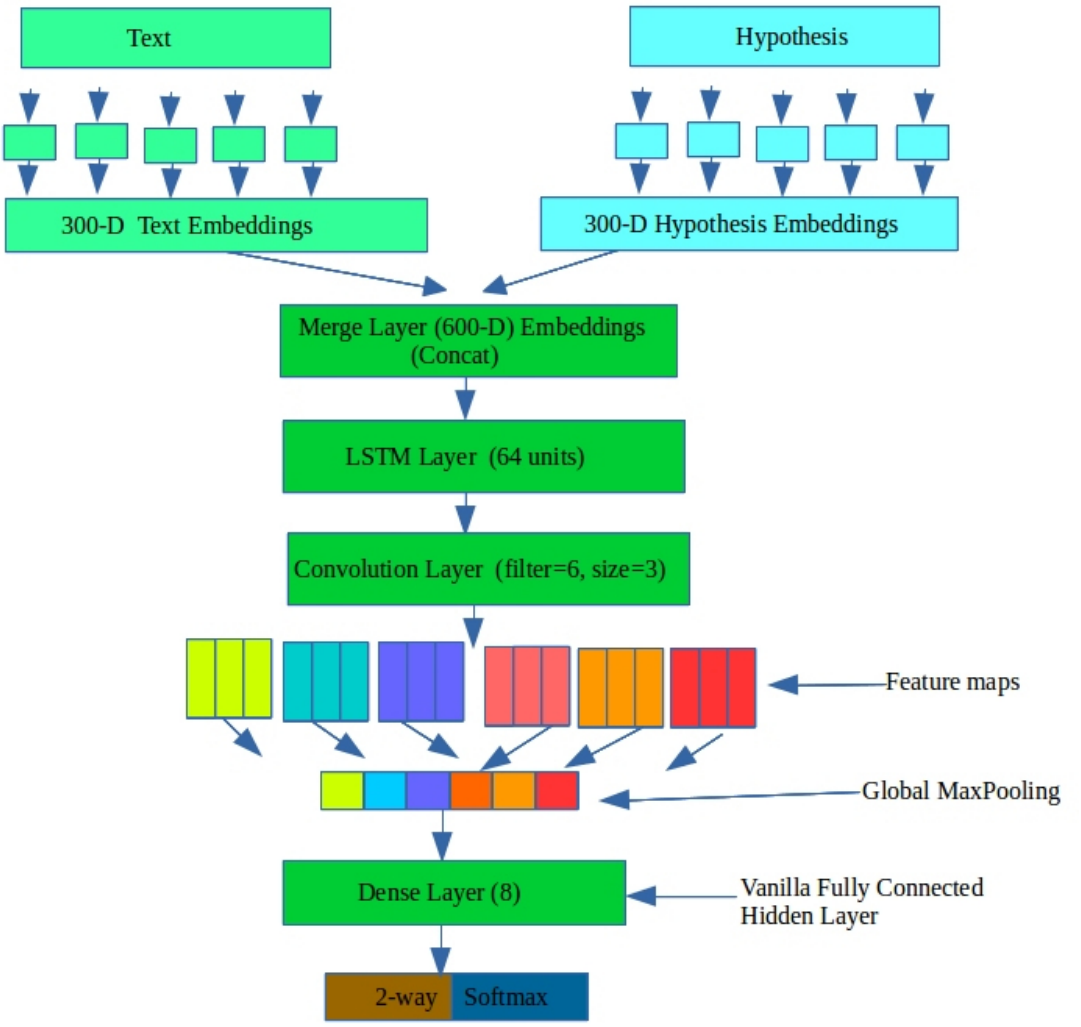

Figure 1: A pictorial illustration of the LSTM-CNN model

in our vocabulary with a vector representation $x_t \in \mathbb{R}^{d=300}$. We generate a sentential representation by performing an element wise sum of each x_t in order to get the sentence embedding for each input sequence, i.e., both for the articles and the queries. We normalize the resulting vector sum by the length of the sequence such that

$$s_i = \frac{1}{|n|} \sum_{t=1}^{|n|} x_t, \quad s_i \in \mathbb{R}^{d=300} \tag{13}$$

Where s_i in equation (13) denotes the embedding representation of each data point.

The encoded sequences are then passed into a Merge layer where the encoded article and query are concatenated. This layer yields a 600 dimensional vector which we pass to the LSTM. The activation function for this layer, like in every other hidden layers of our network is the rectified linear unit (RELU). The LSTM has 64 units, and the representation from the hidden state of the last cell is propagated to a convolution layer. Our convolution layer makes use of 6 filters, each with a region size

= 3. A global maxpooling layer selects the prominent features from the feature maps. We further add a fully connected multilayer perceptron networks (mlp) with 8 hidden layers. We use the softmax to distribute the probability distribution between the +1 or -1 class, where +1 signifies entailment and -1 means Non-entailment. For each input s_j, given the hidden state h_j from the last fully-connected mlp layers, the softmax classifier predicts the label $\hat{y}_j$ as shown in the equation below:

$$\widehat{p}\theta(y|s_j) = softmax\Big(W^{(p)}h_j + b^{(p)}\Big),$$

$$\widehat{y_j} = \arg\max_y \widehat{p}\theta(y|s_j) \tag{14}$$

The cost function is given in equation (15) below:

$$J(\theta) = -\frac{1}{m}\sum_{k=1}^{m}(y^{(k)}|s^{(k)}) + \frac{\lambda}{2}||\theta||_2^2 \tag{15}$$

where m is the number of data samples, in this case the batch size for each iteration and λ is an L2 norm regularization hyperparameter.

For implementation, we used Keras[4] Deep Learning library. Since the training set is rather tiny, we used a batch size of 16. We used *adam*, a stochastic optimizer with learning rate set at 0.01 and a decay value of 1e-4. Throughout the network, we applied a uniform *Dropout* value of 0.2 after each of the LSTM, convolution and the fully connected layers. Initially, the model was trained on the input data for *10* epochs but this was subsequently reduced to *5* since we discovered that training keep exiting at epoch *8* when we used *earlystopping* with patience for unimproved validation loss set to *3*.

4 Results and Analysis

In this section, we discuss the results of both task1 and task2 using the techniques discussed in the previous sections. We evaluate the results on the test set provided by the organizers. We observed from the dataset that there were many cases when multiple articles are retrieved for a single query. Therefore, there was a need to consider the cases where the articles retrieved by our system are a subset of the articles in the gold standard. Hence, we carried out two evaluations for the information retrieval system : strict and lenient. In strict evaluation, only exact matches of the articles retrieved by our system with the gold standard were considered as true positive. In lenient evaluation, a partial match of the retrieved articles with the gold standard set of articles is also considered as a true positive. In other words, if the system retrieves a subset of the articles present in the gold standard then it is considered a match. Table 1 shows the results of task1 for both partial string matching and topic clustering method.

	Partial String Matching (strict evaluation)	Partial String Matching (lenient evaluation)	Topic Clustering (strict evaluation)	Topic Clustering (lenient evaluation)
Precision	0.413	0.645	0.441	0.69
Recall	0.445	0.628	0.49	0.692
F-score	0.428	0.636	0.464	0.691

Table 1: Results for Task 1 (information retrieval)

The results indicate that topic clustering method achieved a higher F-score than the partial string matching method for both strict and lenient evaluation. This is because topic clustering not only relies

[4]Keras is a modular but powerful Deep Learning Library built on top of Theano and Tensorflow. Available at https://github.com/fchollet/keras

on the occurrence of common tokens between article and query but also takes into account the semantics by incorporating topics and a knowledge-based resource (WordNet). We present an example in Figures 2 and 3 to illustrate this. The article retrieved by the partial string matching method captures only a part of the meaning of the query. The query explicitly talks about the conditions in which the contract shall be formed. However, the article retrieved by the partial string matching method talks about some conditions (necessary to form the contract) mentioned in the query but does not mention anything about forming the contract. It captures only the partial meaning of the query. On the other hand, the article retrieved by the topic clustering method mentions both the conditions and forming of the contract. Thus, topic clustering method was successful to capture the semantics of the text to a greater extent as compared to the partial string matching method. The topic clustering method also achieves a higher precision and recall than the partial string matching method. However, it is also important to mention some limitations while using topic clusters for short text similarity. LDA considers the text as a mixture of latent topics. Each topic is a mixture of words. The topics generated in query and article are quite similar when query and article comprise similar words. However, in cases when most words are different in query and article, the topics of the articles (grouped in the cluster) are sometimes not relevant to the topics of the query. We tried to address this limitation to some extent by incorporating synonyms from WordNet. However, even minor variation in the usage of wordings in short texts can lead to different topic distributions. Generally, topic clustering methods have been successful in information retrieval for large corpora which comprise documents of significant length. However, the COLIEE dataset consists of civil codes which are short legal texts.

Query	Article retrieved by Topic Clustering
In the case where (A) loses his/her capacity to act after the dispatch of the notice of acceptance of a contract to (B), who is a person at a distance, even if (B) knows such fact, the contract shall be formed.	(1) A contract between persons at a distance shall be formed upon dispatch of the notice of acceptance. (2) In cases where no notice of acceptance is required due to the offeror's manifestation of intention or usage of trade, the contract shall be formed upon the occurrence of any fact which ought to be regarded as a manifestation of intention of acceptance.

Figure 2: Article retrieved by topic clustering method for a given query

Query	Article retrieved by Partial String Matching
In the case where (A) loses his/her capacity to act after the dispatch of the notice of acceptance of a contract to (B), who is a person at a distance, even if (B) knows such fact, the contract shall be formed.	(1) Manifestation of intention to a person at a distance shall become effective at the time of the arrival of the notice to the other party. (2) The validity of manifestation of intention to a person at a distance shall not be impaired even if the person who made the manifestation dies or loses his/her capacity to act after the dispatch of the notice.

Figure 3: Article retrieved by partial string matching for a given query

Table 2 shows the results of the textual entailment task. We compare our proposed LSTM-CNN model with the best runs of the other teams for the English task. The results show that our model achieves comparable performance with other textual entailment methods. The slightly lower accuracy

Method	Accuracy
JAISTNLP	0.512
JNLP	0.487
iLis	0.576
LSTM - CNN Model	0.538

Table 2: Accuracy for Textual Entailment Task

of our model is probably due to the trained word embeddings on the Google News dataset (which is not tailored for legal text). The results of task 2 indicate that our textual entailment system did not generalize well on the test data. One possible reason for this is the small size of training data in our case. Neural networks have been known to perform well with a large amount of training data. The motivation to use neural networks was to avoid handcrafting features given the legislative nature of the text. Another possible reason for the mediocre performance of the textual entailment system is error propagation due to the evaluation style. This is because the performance of the second system (task 2) is dependent on the results of the first system (task 1). In task 2, we are checking for entailment relationship between each retrieved article and the corresponding query. In the case when the system retrieves wrong articles, the entailment relationship is judged wrongly.

5 Conclusion

This paper presented our team's approach for the COLIEE 2017 competition. For the information retrieval task, we utilized a partial string matching and a topic clustering method. The lexical approach of partial string matching was complemented by a semantic similarity approach of topic clustering while also incorporating the knowledge from WordNet. The results indicate that the topic clustering approach outperformed the partial string matching method. Further, we proposed a LSTM-CNN model for the textual entailment task. We utilized word embeddings from the Google News corpus. The LSTM-CNN model had a comparable performance with other textual entailment systems.

6 Acknowledgments

Research presented in this paper is conducted as a PhD research at the University of Turin, within the Erasmus Mundus Joint International Doctoral (Ph.D.) programme in Law, Science and Technology. Luigi Di Caro and Livio Robaldo have received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Skodowska-Curie grant agreement No 690974 for the project "MIREL: MIning and REasoning with Legal texts".

References

[1] Leif Azzopardi, Mark Girolami, and CJ Van Rijsbergen. Topic based language models for ad hoc information retrieval. In *Neural Networks, 2004. Proceedings. 2004 IEEE International Joint Conference on*, volume 4, pages 3281–3286. IEEE, 2004.

[2] Petr Baudiš and Jan Šedivỳ. Sentence pair scoring: Towards unified framework for text comprehension. *arXiv preprint arXiv:1603.06127*, 2016.

[3] Luisa Bentivogli, Peter Clark, Ido Dagan, Hoa Dang, and Danilo Giampiccolo. The seventh pascal recognizing textual entailment challenge. *Proceedings of TAC*, 2011, 2011.

[4] David M Blei and John D Lafferty. Dynamic topic models. In *Proceedings of the 23rd international conference on Machine learning*, pages 113–120. ACM, 2006.

[5] David M Blei, Andrew Y Ng, and Michael I Jordan. Latent dirichlet allocation. *the Journal of machine Learning research*, 3:993–1022, 2003.

[6] 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.

[7] Scott Deerwester, Susan T Dumais, George W Furnas, Thomas K Landauer, and Richard Harshman. Indexing by latent semantic analysis. *Journal of the American society for information science*, 41(6):391, 1990.

[8] Miguel Angel Ríos Gaona, Alexander Gelbukh, and Sivaji Bandyopadhyay. Recognizing textual entailment using a machine learning approach. In *Mexican International Conference on Artificial Intelligence*, pages 177–185. Springer, 2010.

[9] Wael H Gomaa and Aly A Fahmy. A survey of text similarity approaches. *International Journal of Computer Applications*, 68(13), 2013.

[10] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[11] Thomas Hofmann. Probabilistic latent semantic indexing. In *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 50–57. ACM, 1999.

[12] Sergio Jimenez, George Duenas, Julia Baquero, Alexander Gelbukh, Av Juan Dios Bátiz, and Av Mendizábal. Unal-nlp: Combining soft cardinality features for semantic textual similarity, relatedness and entailment. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 732–742, 2014.

[13] Adebayo Kolawole John, Luigi Di Caro, and Guido Boella. Normas at semeval-2016 task 1: Semsim: A multi-feature approach to semantic text similarity. *Proceedings of SemEval*, pages 718–725, 2016.

[14] Adebayo Kolawole John, Luigi Di Caro, Guido Boella, and Cesare Bartolini. An approach to information retrieval and question answering in the legal domain.

[15] Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. A convolutional neural network for modelling sentences. *arXiv preprint arXiv:1404.2188*, 2014.

[16] Tom Kenter and Maarten de Rijke. Short text similarity with word embeddings. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, pages 1411–1420. ACM, 2015.

[17] Mi-Young Kim, Ying Xu, and Randy Goebel. A convolutional neural network in legal question answering.

[18] Yoon Kim. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*, 2014.

[19] Thomas K Landauer, Peter W Foltz, and Darrell Laham. An introduction to latent semantic analysis. *Discourse processes*, 25(2-3):259–284, 1998.

[20] Yuhua Li, David McLean, Zuhair A Bandar, James D O'shea, and Keeley Crockett. Sentence similarity based on semantic nets and corpus statistics. *Knowledge and Data Engineering, IEEE Transactions on*, 18(8):1138–1150, 2006.

[21] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[22] Shachar Mirkin, Ido Dagan, and Eyal Shnarch. Evaluating the inferential utility of lexical-semantic resources. In *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics*, pages 558–566. Association for Computational Linguistics, 2009.

[23] Tim Rocktäschel, Edward Grefenstette, Karl Moritz Hermann, Tomáš Kočiskỳ, and Phil Blunsom. Reasoning about entailment with neural attention. *arXiv preprint arXiv:1509.06664*, 2015.

[24] Jiannan Wang, Guoliang Li, and Jianhua Fe. Fast-join: An efficient method for fuzzy token matching based string similarity join. In *Data Engineering (ICDE), 2011 IEEE 27th International*

Conference on, pages 458–469. IEEE, 2011.