AN APPROACH TO INFORMATION RETRIEVAL AND QUESTION ANSWERING IN THE LEGAL DOMAIN

Adebayo Kolawole John, Luigi Di Caro, Guido Boella, and Cesare Bartolini

Dipartimento di Informatica, Universita Di Torino
Corso Svizzera 185, Torino, 10149, Italy
{kolawolejohn.adebayo}@unibo.it, {dicaro,guido.boella}@di.unito.it,
{cesare.bartolini}@uni.lu,

Abstract. We describe in this paper, a report of our participation at COLIEE 2016 Information Retrieval (IR) and Legal Question Answering (LQA) tasks. Our solution for the IR part employs the use of a simple but effective Machine Learning (ML) procedure. Our Question Answering solution answers "YES or 'NO' to a question, i.e., 'YES' if the question is entailed by a text and 'NO' otherwise. With recent exploit of Multi-layered Neural Network systems at language modeling tasks, we presented a Deep Learning approach which uses an adaptive variant of the Long-Short Term Memory (LSTM), i.e. the Child Sum Tree LSTM (CST-LSTM) algorithm that we modified to suit our purpose. Additionally, we benchmarked this approach by handcrafting features for two popular ML algorithms, i.e., the Support Vector Machine (SVM) and the Random Forest (RF) algorithms. Even though we used some features that have performed well from similar works, we also introduced some semantic features for performance improvement. We used the results from these two algorithms as the baseline for our CST-LSTM algorithm. All evaluation was done on the COLIEE 2015 training and test sets. The overall result confirms the competitiveness of our approach.

Keywords: Child-Sum tree lstm, Information Retrieval, textual entailment

1 Introduction

COLIEE is an annual competition[1] for legal Information Extraction (IE) and Retrieval. COLIEE 2016 focuses on two main tasks, i.e. Textual Entailment (TE) and Information Retrieval (IR) all proposed as a form of Legal question answering task. The organizers divided the task into three sub-tasks, i.e., subtask one involves reading a legal bar exam question Q, presented as a query, and extracting a subset of the Japanese Civil Code Articles C_1,C_2,C_3...,C_n from the entire Civil Codes made available in the COLIEE Dataset. The set of civil codes retrieved by the system must sufficiently answer the initial query Q.

[1] collocated with the JURISIN workshop

Sub-Task two advances this objective by identifying entailment relationship between an article or a set of articles and a given query, e.g., $E_f(C_1, C_2, C_3..., C_n, Q)$, where E_f is the entailment function that produces a True or False answer. The third combines the efforts of the first and second tasks. For the tasks, the organizers released a corpus of Japanese Legal Civil Code Articles, the articles have been translated into English for obvious reasons.

This paper presents a report of our participation in the first and second tasks, as well as a description of the systems we developed for these tasks. The remaining parts of the paper is structured as follows. In section 2, we succinctly describe the general IR procedure and our own approach. Next, we describe the TE problem and the state of the arts. This is followed by elucidation of the features used for the SVM and RF classification and subsequently the Deep Learning approach presented. Finally, we discuss the Experiments and the results obtained.

2 Task 1- Information Retrieval

Information Retrieval (IR) involves retrieving a set of item(s), usually from among multiple options such that the result satisfies a user's information need (Query). Earliest approach to IR involves the use of Lexical Matching or Keyword Search. This is however grossly inefficient as it is not robust to Natural Language ambiguities (i.e., synonymy and polysemy). The Bag of Word (BOW) based models such as LSA [7] and Term Frequency Inverse Document Frequency (TFIDF) partially overcome these ambiguities by grouping terms that occur in a similar context into the same semantic cluster based on their vector representation in the semantic space. Topic Modeling approaches have also been used [17, 1]. The idea here is to probabilistically assign topics to documents as well as query based on their term composition. Similar documents tend to belong to similar topics, thus, the task would be to retrieve documents with similar topics to the query. LDA [4], a generative probabilistic algorithm excel in this category. Since the Query is usually a fraction of the document, a common approach for improving retrieval accuracy is by Query Expansion which uses the Part-of-Speech (POS) to retrieve synonyms for Query enrichment. While this partially helps, the approaches still do not necessarily capture the full semantics of a language since they discard useful information about word order and context without word-sense disambiguation. Recently, Machine Learning techniques are being employed to overcome this barrier, e.g., the authors in [8, 30] proposed Deep Learning (DL) architectures. There are now evolving some IR systems which can arguably be classified as Question Answering (QA) systems. These systems exploit the massive set of structured information[2] available on some Knowledge Base (KB) like the DBPedia, Freebase, Wikipedia and BabelNet etc., to retrieve information about any Query. QAKIS [6] and DEQA [18]

[2] These information are serialized as structured SPO triples e.g., Subject-Predicate-Object facts.

are examples of these systems and should not be confused with the demand of the underlying task.

While encouraging results have been reported with DL techniques, these systems generalize better when trained on a large dataset. However, COLIEE training data is different in that the set of documents to be queried are not documents in actual sense but Japanese Civil Codes, each containing less than 4 sentences. Also, the total number of articles available is very small, actually less than 500 instances. Our idea is that a simple challenge should not be approached with a rather too complex system. More so, simple solutions in NLP (e.g. N-gram models) often compete favourably with some complex techniques. We submitted three runs using simple machine learning approaches. We describe our solution below.

2.1 Distributed Representation and Information Retrieval

It can be argued that IR is about finding semantic connection between a query and some document(s) in a corpus. In our case, the corpus to be queried is the set of Japanese Civil Codes. We limit ourselves to finding the most related of the civil codes to the query. As earlier explained, modeling language as BOW trades away several useful contextual information which Humans find useful in aggregating meaning. An obvious choice is to take advantage of the distributed representation of words. Distributional Semantics rely on the idea that words within proximity usually have similar meaning [32]. With application of Neural Networks, it is now possible for language models to learn high dimensional representation of words and their relationship such that semantically related words are transformed into similar vector representations. Specifically, such representation have greatly shown to capture semantic relatedness and similarity, thus overcoming sparsity problems in atomic representation. The Word2Vec[3] [22] is a popular algorithm for inducing such vector space representations of words, commonly known as word embeddings. The method exploits the Distributional Hypothesis [32] and works best when trained on large corpora. Furthermore, algebraic computations could be performed on the embeddings, which still retain the latent semantic characteristics of the vectors, e.g., *Man* + *King* - *Woman* = *Queen*. Also, relying on compositionality[4] of words [10], atomic vectors could be combined to generate phrasal or sentential representation. This ensures that the meaning of words are not captured in isolation, since the true meaning of a sentence must take into account the interplay between the words it contains [24, 25, 10].

Our task is thus to generate a sentential representation for all the Articles as well as queries, with an hypothetical assumption that sentential vectors of semantically related articles and queries would point to the same direction. In other words, we simply create a mapping between the Article(s) representation and the query representation. Since these representations are vectors, a simple

[3] Word2vec is available at https://code.google.com/p/word2vec/

[4] e.g., *man* + *marry* + *woman* is closer to the vectors of *child* than *car*.

choice is to measure the distance between that of a query and each Article with metrics like the Cosine Similarity and then rank the scores. However, this may not generalize as expected. Without loss of generality, our embedding based approach is able to capture the relatedness between embeddings of query to some document even with little feature generation.

First, we describe how to generate the word embedding. we used a combination of the training and test data as well as 19,348 documents from Eur-Lex dataset[5]. There is no specificity in the choice other than we needed a fair collection of legal text to train the Word2Vec algorithm on. Moreover, this is still by order of magnitude a very small amount of data for proper distributed representation. We used the Gensim[6] implementation of the algorithm.[7] Gensim implements a variant of the algorithm known as *Skip-Gram,* which trains word representations using a model that when given a word will predict what words are likely to occur within a fixed window around it.[8]

2.2 Model Description

Assume that each Article (hence Document) contains words x_i,x_{i+1},x_{i+2},x_{i+3}...x_n. We associate each word w in our vocabulary with a vector representation $\mathrm{x}_w \in \mathbb{R}^d$. Each x_w is of dimension $d \times V$ of the word embedding matrix W_e, where V is the size of the vocabulary. For each Document D, we generate a representation by performing an element wise sum of each $\mathrm{x}_w \in$ D. We normalize the resulting vector sum by the length of the sequence such that

$$s_i = \frac{1}{|n|} \sum_{i=1}^{|n|} x_i, \quad s_i \in \mathbb{R}^{d \times V} \tag{1}$$

where s_i denotes the embedding representation of a sentence. In the COLIEE dataset for subtask 1, each Document is associated to a Label. Actually, since the original XML document contains pairs of tag *t1* (article) and *t2* (hypothesis), it is possible to have more than one article within a *t1* tag. In this case, we exclusively separate each article as an independent document along with their subheading and labels as below. The number and phrase in bold are the label and the title respectively, followed by the main text:

265 Content of Superficies *A superficiary shall have the right to use the land of others in order to own structures, or trees or bamboo, on that land.*

[5] http://www.ke.tu-darmstadt.de/resources/eurlex

[6] Gensim is a python library for an array of NLP tasks. It is available at https://radimrehurek.com/gensim/

[7] Parameters used: Context Window: 5, Neural Network layer size: 50, Minimum word count: 5.

[8] Skip-gram training generally performs better than an alternative Word2Vec model known as Continuous-Bag-of-Words (CBOW) that uses an alternative objective that tries to predict a word by conditioning on all of the words that surround it within a window.

As the titles carry important information, we combined them with the main text to form each complete article.

Based on our observation of the training sample, none of the articles have more than one unique label, We therefore approach the problem as a modest multi-class classification task, i.e., Given a training data set of the form (d_i,y_i), where $\mathrm{d}_i \in \mathbb{R}^D$ is the ith example and $\mathrm{y}_i \in \{1, \ldots, \mathrm{K}\}$ is the ith class label, we aim at finding a learning model $\mathbb{H}$ such that $\mathbb{H}(\mathrm{q}_i) = \mathrm{y}_i$, where an unseen example $\mathrm{q}_i \in \mathbb{R}^Q$ is a query instance which by our assumption correlates to a document d_i and Q is the set of all the queries. Our goal is for $\mathbb{H}$ to capture semantic relatedness between (d_i, q_i).

In order to avoid excessive feature generation, we use the sentential representation, s_i of each article. Recall that this representation is a sum over all individual word vectors x_w for each document d_i. Additional, we use 4 simple features used to capture the semantic distance of each article to another. To do this, we obtained a corpus representation D_c, where like we did for each document, D_c is a sum over all the vectors of each $\mathrm{d}_i \in \mathbb{R}^D$. Likewise, we obtained a representation Q_c for all the Queries. Using cosine similarity, we measure the distance between each article and D_c as well as between each article and Q_c. The first is to capture how similar each article is to the semantic space of all articles while the second captures the gap to the semantic space of all queries. Also, using cosine similarity, we compare s_i of each article to other articles, we rank and pick the top two most similar articles to it. We use the similarity scores of these top similar articles as the third and fourth features. We feed into our classifier, the embedding s_i of each document along with these four features appended. To the best of our knowledge, we did not encounter any works that have used word embeddings for classification as we have done here.

For the first two runs, we used the Scikit implementation of Support vector Machine (SVM) and Random Forest (RF) algorithms as the choice classifiers. We used Grid-Search to find the optimal set of parameters.

2.3 Word Clustering

Our baseline uses a flat clustering algorithm, K-Means [11]. Only constrained by the number of clusters K which must be set a priori, it maintains that number of so-called cluster centroids. The idea is to group related words based on their distribution in the corpus. Usually, similar documents tend to have similar word distribution and by extension, cluster distribution. The LDA [4] would have been a natural choice, being generative. However, we are constrained by the rather tiny size of the data we are working with, especially since LDA performs best with big training data. We used K-Means cluster size of 15. The algorithm was fed with a BOW representation of the training documents which was then used to transform the queries into their cluster representations. The idea is that instead of comparing the cluster distribution for a query with that of all documents in the training set, we simply compare with those in the same class and then calculate the pairwise distance of their vectors with cosine similarity.

3 Task 2- Textual Entailment

The sub-task2 is a Textual Entailment (TE) task. TE aims at determining whether an hypothesis h is entailed by a text t [3]. In fact, most semantic representation tasks in NLP can be reduced to a TE task, e.g., text similarity [14]. Previous works approached TE as a classification task, using hand crafted features [9] and over relying on knowledge resources [23] e.g., WordNet, etc,. At times, these features could be as simple as n-gram overlap, string matching, presence of negation words e.g., never, shouldn't etc., as well as IR techniques such as TFIDF and cosine similarity [13]. The authors in [16] recently employed a Convolutional Neural Network (CNN) to model entailment relationship on COLIEE dataset, we use their system as a baseline for our results. We attempted the same problem with two different approaches, i.e., conventional ML with feature engineering and Deep Learning (DL) technique. For our ML solution, following similar works, we handcrafted some features typical of measuring similarity between two text snippets., while in the latter, we rely on the ability of neural networks to autonomously learn latent features from data for good generalization. We explain the two systems below. TE could be approached either as a binary classification(YES/NO) or multi-class classification (YES/NO/NEUTRAL). The organizers proposed a binary classification task.

3.1 Machine Learning Approach

We obtained the following features from the t and h in order to train the algorithms. The features include common similarity features, some semantic features that we introduced as well as a negation feature which captures the occurrence of words with negative sentiments e.g., Can't, Should't etc. We combined the features and fit into their labels using the RF and SVM algorithms. As in the previous experiment, we use grid-search for parameter optimization.
String matching:We compare each t and h character by character and obtained the Longest common Substring, Levenshtein Distance as well as Jaccard Similarity calculated as

$$Jac = \frac{t \cap h}{t \cup h} \tag{2}$$

Word Ordering: We use the union of all tokens in a pair of sentences to build a vocabulary of non-repeating terms. Building a vector for each sentence, from their position mapping in the vocabulary. If a vocabulary term is found in the sentence, the index number of that term in the vocabulary is added to the vector. Otherwise, similarity of the vocabulary term and each term in the sentence is calculated using a WordNet based word similarity algorithm. The index number of the sentence term with highest similarity score above a threshold is added. If the first two conditions does not hold, 0 is added to the vector.
Word Overlap: We use the word n-gram overlap feature [29]. The n-grams overlap is defined as the harmonic mean of the degree of mappings between the first and second sentence and vice versa, requiring an exact string match of

n-grams in the two sentences.

$$\text{Ng(A,B)} = \left(2(\frac{\mid A \mid}{A \cap B} + \frac{\mid B \mid}{A \cap B})^{-1}\right) \tag{3}$$

Where A and B are the set of n-grams in the sentence and hypothesis. We computed three separate features using equation 2 for each of the following character n-grams: unigram, bigrams and trigrams. We also use weighted word overlap which uses information content [27].

$$\text{wwo(A,B)} = \frac{\sum_{w \in A \cap B} ic(w)}{\sum_{w' \in B} ic(w')} \tag{4}$$

$$\text{ic(w)} = \left(\ln \frac{\sum_{w' \in C} freq(w')}{freq(w)}\right) \tag{5}$$

Where C is the set of words and *freq(w)* is the occurrence count obtained from the Brown corpus. Our weighted word overlap feature is computed as the harmonic mean of the functions *wwo(A,B)* and *wwo(B,A)*.

POS Overlap: We used the Stanford POS-tagger to tag the words in each sentence with their POS categories. We group the words by the following coarse grained POS categories: nouns, verbs, adjectives and adverbs. We then compare the overlap between these classes of part of speech in each sentence. The POS overlap features are defined as the Jaccard similarity of the two sentences across just the words in a particular POS category:

$$POV_{pos} = \frac{|A_{pos} \cap B_{pos}|}{|A_{pos} \cup B_{pos}|} \tag{6}$$

Where A_{pos} and B_{pos} are the set of terms in POS class *pos* of sentences 1 and 2, respectively. This results in 4 unique POS overlap features.

Dependency Parsing: We used *Stanford Parser* [20] to extract the *subject-verb-object* triples from each sentence. We compare the *subject* and *object* of both sentences. If any of the objects or subjects in a sentence is a NE, we simply compare with the corresponding one from the other sentence by pure string matching. When the same word takes either the role of *subject* or *object* in the two sentences, we assign a flat score of 0.5. If both the *subject* and *object* in the two sentences correspond to the same NEs as in the example above, we assign a score of 1.0. Otherwise, we assign a score of 0.0.

Word Embedding Similarity: Using the embedding from the Word2Vec model trained earlier, we obtained the cosine of the semantic representation of both the text and hypothesis. Where the semantic representation is an additive and multiplicative composition of words in each sentence.

WordNet similarity: Based on the ideas in [19], we obtained WordNet similarity between a text and hypothesis relying both on the path length as well as the depth function.

Negation Words: We use the presence of negation words to determine if entailment exist. First, we curated a list of negation words and also a dictionary of

anti-negation words e.g., *nobody:somebody, no one:someone, nothing:something, no where:somewhere, neither:either* etc, each well mapped to its respective negation word. We normalized all shortened form of a negation word to its true form e.g., *couldn't -> could not* etc. In all, we had 22 words. For any negation word found in t, we look for the same in h and viz versa. Using dependency parsing, we compare the role of such word in both. If they both have the same semantic role, we assign a fixed score of 1 and 0 otherwise. Secondly, for any negation word found in h, we look for the presence of its respective anti-negation word from the dictionary in t and also if they bear the same dependency roles, we assign a score 1, otherwise 0. Lastly, we compare the ratio of negation words in t to h, this behaves like overlap measure.

Vector Space based Similarity: We used the feature extraction module of the *scikit-learn* to extract the *TFIDF* weighted feature vectors for the two sentences and then calculated the cosine similarity between them:

$$\cos(A, B) = \frac{\sum_{t=1}^{n} TFIDF(w_{t,A})TFIDF(w_{t,B})}{\sqrt{\sum_{t=1}^{n} w_{t,A}^2 \sum_{t=1}^{n} w_{t,B}^2}} \tag{7}$$

Where A and B are the text and hypothesis.

3.2 Deep Learning Approach

Recently, researchers using algorithms like the CNN [15], Recurrent Neural Networks (RNN) [21], Long Short-Term Memory (LSTM) [12] have reported encouraging results on Language Modeling tasks. Most importantly, [5, 28, 2] showed that deep Neural architecture can match and outperform lexical classifiers which use hand-crafted features for TE. Inspired by these results, we employed a variant of Tree LSTM, the Child-Sum Tree LSTM proposed by Tai et al., [31] and which was previously applied for semantic relatedness task. Tree-LSTMs are specialized LSTMs that adopt the tree-structure topology, i.e., at any given time step t the LSTM is able to compose its states from input vector and hidden states of its child-nodes simultaneously. This is unlike the standard LSTM that assumes a single child per unit. The Child-Sum-Tree LSTMs transition is represented by the following equation:

$$\widehat{h_j} = \sum_{k \in C_{(j)}} h_k \tag{8}$$

$$i_j = \sigma\left(W^{(i)}x_j + U^{(i)}\widehat{h_j} + b^{(i)}\right) \tag{9}$$

$$f_j k = \sigma\left(W^{(f)}x_j + U^{(f)}\widehat{h_k} + b^{(f)}\right) \tag{10}$$

$$o_j = \sigma\left(W^{(o)}x_j + U^{(o)}\widehat{h_j} + b^{(o)}\right) \tag{11}$$

$$u_j = \tanh\left(W^{(u)}x_j + U^{(u)}\widehat{h_j} + b^{(u)}\right) \tag{12}$$

$$c_j = i_j \odot u_j + \sum_{k \in C_{(j)}} f_{jk} \odot c_k \tag{13}$$

$$h_j = o_j \odot \tanh c_j \tag{14}$$

C_j are the child-nodes in unit j and k $\in$ C_j. In order to generate a semantic representation for a pair of text and hypothesis, each child of a tree is a node from a dependency parse tree of a text or hypothesis as well as its child nodes. For each node, the algorithm takes as input the word vectors. We obtain a representation (h_{txt}, h_{hyp}) from both the text and hypothesis considering the distance and angle of their element-wise summed vectors as well multiplied vectors. We use Neural Network to predict the probability of either +1 or -1 class, where +1 signifies entailment and -1 means Non-entailment. At each node j, we use a softmax classifier to predict the label $\hat{y}_j$ given the inputs x_j observed at nodes in the subtree rooted at j. The classifier takes the hidden state h_j at the node as input, considering both the distance and angle between the pair as shown in the equations below:

$$h_\times = h_{txt} \odot h_{hyp},$$

$$h_+ = h_{txt} + h_{hyp},$$

$$h_\diagdown = |h_{txt} - h_{hyp}|,$$

$$h_s = \sigma\left(W^{(\times)} h_\times + W^{(\times)} h_\diagdown + W^{(+)} h_+ + b^{(h)}\right),$$

$$\widehat{p\theta} = softmax\left(W^{(p)} h_s + b^{(p)}\right),$$

$$\widehat{y_j} = \arg\max_y \widehat{p\theta}(y|x_j) \tag{15}$$

The cost function is given below:

$$J(\theta) = -\frac{1}{m} \sum_{k=1}^{m} (y^{(k)}|x^{(k)}) + \frac{\lambda}{2} ||\theta||_2^2 \tag{16}$$

where m = 2, i.e., (YES/NO) and is an L2 norm regularization hyperparameter.

4 Experiments

The COLIEE corpus contains pairs of sentences, i.e., the civil code article well labeled with an identifier, e.g., 'Article 572' as well as the query question. The Query question is ignored for the first task as it serves as the hypothesis for entailment in the second task. In all, 296 civil code articles, each with distinct ID was released as training data for task one while 490 pairs of civil code articles an query was made available as training data for the second task. Furthermore,

task 1 has 95 test data while task 2 has 66 test data. Since the organizers did not release the official Gold standard for the tasks and/or the official performance evaluation of the submitted systems, we only report an evaluation on the 2015 dataset.

Due to the setup of our IR system, more attention is focused on precision than recall. For instance, since we approached it as a classification task, the system only predict an article per query, i.e. even if a query has more than one related articles, the model only select the one with highest confidence score. In a way, this can also reduce recall but many unrelated articles can be avoided from the result set. For the purpose of evaluation, we simply use accuracy score, which is calculated as the number of correct prediction over the total number of prediction, i.e., if out of 10 queries, the system got 4 correctly, then the accuracy is 0.40. Tables 1 and 3 summarize the results obtained on task 1 and 2. In particular, Table 3 shows the result on a 5-fold cross validation run for the second subtask. As shown in table 3, we used the result reported by Kim et al., [16] as our baseline score since their evaluation was on the same COLIEE data. We observe a slightly better performance from the DL approach which even outperforms the baseline system. We performed an ablation test on the features designed in order to see the best mix. To do this, we made 3 runs by randomly withholding 3 features per time. The table 2 shows the performance of the SVM for each runs.

Model	No of Train Article	No of Test Query	Accuracy
Run1 (SVM)	296	79	0.521
Run2 (RF)	296	79	0.560
Run3 (KM)	296	79	0.481

Table 1. Task 1 Evaluation

Model	Withheld Features	Accuracy
Run1 (SVM)	Word Embedding Similarity, POS overlap, Dependency parsing	0.460
Run2 (SVM)	Negation Words 1 and 2, word overlap	0.51
Run3 (SVM)	Negation Words 2, Word Embedding Similarity, Vector space similarity	0.394

Table 2. Feature Ablation Evaluation on Task 1

Model	Train	Test
SVM (Features+)	78.20	68.40
RF (Features+)	76.00	64.19
Child-Sum-Tree LSTM	81.33	70.10
Baseline (Kim et al.,) [16]	-	55.87

Table 3. Task 2 Evaluation on COLIEE dataset

We implemented a child-Sum Tree LSTM proposed by [31]. As earlier stated, we used our pre-trained word embeddings as input vector to the LSTM, instead of one-hot encoding representation with sequence to sequence learning. Our embedding has 50 dimensions.

For implementation, we used the Keras[9] Deep Learning library . Our LSTM model has an hidden layer size of 300. We used a batch-size of 25. We used ADAM, a stochastic optimizer with learning rate set at 0.01 and a decay value of 1e-4 as well as momentum of 0.9. Typically, we set the number of training epochs to 200. However, throughout the experiments, we track the model for over-fitting by monitoring the peak accuracy on validation data as well as the epoch where the model does not seem to learn sufficiently again and accuracy begins to drop or does not increase anymore. Usually, we pick this epoch for our test runs. Since the organizers did not include any validation set, we divided the test data into two. We actually tested on half while the other half was used as the validation set. As observable, our system is able to generalize even with little training data. It should be noted that the quality of embedding used as input also influences the result obtained. However, we did not verify how the use of a bigger pretrained embedding like the Glove vectors [26] can improve performance

5 Conclusion

The paper described our team's attempt at solving some of the challenges proposed by COLIEE2016 organizers. We participated in two tasks, i.e. IR and TE which is receiving lots of attention in NLP domain generally. For the first task, we submitted 3 runs, each based on SVM, RF and K-means clustering which was used as the baseline. In the second task, we designed some novel features combined and fit into the labels with SVM and RF algorithms. Secondly, we adapted the Child-sum tree LSTM for the textual entailment task. The result obtained on a 5-fold cross validation shows that the DL system slightly outperformed the SVM and RF systems.

Acknowledgments. Kolawole J. Adebayo has received funding from the Erasmus Mundus Joint International Doctoral (Ph.D.) programme in Law, Science and Technology. Luigi Di Caro and Guido Boella have received funding from the European Union's H2020 research and innovation programme under the 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.

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

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, Andrew Y Ng, and Michael I Jordan. Latent dirichlet allocation. *Journal of machine Learning research*, 3(Jan):993–1022, 2003.
5. 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.
6. Elena Cabrio, Julien Cojan, Alessio Palmero Aprosio, Bernardo Magnini, Alberto Lavelli, and Fabien Gandon. Qakis: an open domain qa system based on relational patterns. In *Proceedings of the 2012th International Conference on Posters & Demonstrations Track-Volume 914*, pages 9–12. CEUR-WS. org, 2012.
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. Jianfeng Gao, Li Deng, Michael Gamon, Xiaodong He, and Patrick Pantel. Modeling interestingness with deep neural networks, June 13 2014. US Patent App. 14/304,863.
9. 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.
10. Edward Grefenstette, Mehrnoosh Sadrzadeh, Stephen Clark, Bob Coecke, and Stephen Pulman. Concrete sentence spaces for compositional distributional models of meaning. In *Computing Meaning*, pages 71–86. Springer, 2014.
11. John A Hartigan and Manchek A Wong. Algorithm as 136: A k-means clustering algorithm. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 28(1):100–108, 1979.
12. Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.
13. 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.
14. 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.
15. Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. A convolutional neural network for modelling sentences. *arXiv preprint arXiv:1404.2188*, 2014.
16. Mi-Young Kim, Ying Xu, and Randy Goebel. A convolutional neural network in legal question answering.
17. 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.
18. Jens Lehmann, Tim Furche, Giovanni Grasso, Axel-Cyrille Ngonga Ngomo, Christian Schallhart, Andrew Sellers, Christina Unger, Lorenz Bühmann, Daniel Gerber, Konrad Höffner, et al. Deqa: deep web extraction for question answering. In *International Semantic Web Conference*, pages 131–147. Springer, 2012.

19. 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.
20. Christopher D Manning, Mihai Surdeanu, John Bauer, Jenny Rose Finkel, Steven Bethard, and David McClosky. The stanford corenlp natural language processing toolkit. In *ACL (System Demonstrations)*, pages 55–60, 2014.
21. LR Medsker and LC Jain. Recurrent neural networks. *Design and Applications*, 2001.
22. Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.
23. 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.
24. Jeff Mitchell and Mirella Lapata. Vector-based models of semantic composition. In *ACL*, pages 236–244, 2008.
25. Jeff Mitchell and Mirella Lapata. Language models based on semantic composition. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 1-Volume 1*, pages 430–439. Association for Computational Linguistics, 2009.
26. Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. In *EMNLP*, volume 14, pages 1532–43, 2014.
27. Philip Resnik. Using information content to evaluate semantic similarity in a taxonomy. *arXiv preprint cmp-lg/9511007*, 1995.
28. Tim Rocktäschel, Edward Grefenstette, Karl Moritz Hermann, Tomáš Kočiskỳ, and Phil Blunsom. Reasoning about entailment with neural attention. *arXiv preprint arXiv:1509.06664*, 2015.
29. Frane Šarić, Goran Glavaš, Mladen Karan, Jan Šnajder, and Bojana Dalbelo Bašić. Takelab: Systems for measuring semantic text similarity. In *Proceedings of the First Joint Conference on Lexical and Computational Semantics-Volume 1: Proceedings of the main conference and the shared task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation*, pages 441–448. Association for Computational Linguistics, 2012.
30. Yelong Shen, Xiaodong He, Jianfeng Gao, Li Deng, and Grégoire Mesnil. A latent semantic model with convolutional-pooling structure for information retrieval. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*, pages 101–110. ACM, 2014.
31. Kai Sheng Tai, Richard Socher, and Christopher D Manning. Improved semantic representations from tree-structured long short-term memory networks. *arXiv preprint arXiv:1503.00075*, 2015.
32. Peter D Turney, Patrick Pantel, et al. From frequency to meaning: Vector space models of semantics. *Journal of artificial intelligence research*, 37(1):141–188, 2010.