Legal Question Answering Using Paraphrasing and Entailment Analysis

Mi-Young Kim[1], Ying Xu[1], Yao Lu[2], and Randy Goebel[1]

Dept. of Computing Science, University of Alberta, Edmonton, AB, Canada[1]
iLab Tongji, School of Software Engineering, Tongji University[2]

{ miyoung2,yx2,rgoebel}@ualberta.ca[1]
95luyao@tongji.edu.cn[2]

Abstract. Our legal question answering system combines legal information retrieval and textual entailment, and we describe a system that exploits paraphrasing and sentence-level analysis of queries and legal statutes. We have evaluated our system using the training data from the competition on legal information extraction/entailment (COLIEE)-2016. The competition focuses on the legal information processing required to answer yes/no questions from Japanese legal bar exams, and it consists of two phases: legal ad-hoc information retrieval (Phase 1), and textual entailment (Phase 2). Phase 1 requires the identification of Japan civil law articles relevant to a legal bar exam query. For this phase, we have used an information retrieval approach using TF-IDF and a Ranking SVM. Phase 2 requires decision on yes/no answer for previously unseen queries, which we approach by comparing the approximate meanings of queries with relevant articles. Our meaning extraction process uses a selection of features based on a kind of paraphrase, coupled with a condition/conclusion/exception analysis of articles and queries. We also identify synonym relations using word embedding, and detect negation patterns from the articles. Our heuristic selection of attributes is used to build an SVM model, which provides the basis for making a decision on the yes/no questions. Experimental evaluation demonstrates the value of our method, and the results show that our method outperforms previous methods.

Keywords: legal question answering, recognizing textual entailment, information retrieval, paraphrasing

1. Task Description

Our approach to legal question answering combines information retrieval and textual entailment. We achieve this combination with a number of intermediate steps. For instance, consider the question "Is it true that a special provision that releases warranty can be made, but in that situation, when there are rights that the seller establishes on his/her own for a third party, the seller is not released of warranty." A system must first identify and retrieve relevant documents (typically legal statutes), and subsequently, identify a most relevant sentence. Finally it must extract and

compare semantic connections between the question and the relevant sentences, and confirm a threshold of evidence about whether an entailment relation holds.

The Competition on Legal Information Extraction/Entailment (COLIEE) 2016[1] focuses on two aspects of legal information processing related to answering yes/no questions from legal bar exams: a legal document retrieval (Phase 1), and whether there is a textual entailment relation between a query and relevant legal documents (Phase 2).

We treat Phase 1 as an ad-hoc information retrieval (IR) task. The goal is to retrieve relevant Japan civil law statutes or articles that are related to a question in legal bar exams, from which we can confirm a yes or no answer based on deciding if there is an entailment relation between the question (or the negation of the question) and the relevant statutes.

We approach the information retrieval part of this problem (Phase 1) with two models based on statistical information. One is the TF-IDF model [1], i.e., term frequency-inverse document frequency. The idea is that relevance between a query and a document depends on their intersecting word set. The importance of words is measured with a function of term frequency and document frequency as parameters. Our terms are lemmatized words, which mean verbs like "attending," "attends," and "attended" are lemmatized as the same form "attend."

Another popular model for text retrieval is a Ranking SVM model [2]. We use that model to re-rank documents that are retrieved by the TF-IDF model. The model's features 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 heuristically confirming entailment of a query (or its negation) from relevant articles. The answer to a question is typically determined by measuring some kind of semantic similarity between question and answer. Because the legal bar exam query and relevant articles are complex and varied, we need to carefully determine what kind of information is needed for confirming textual entailment. Here we exploit a kind of paraphrasing based on term expansion and word embedding for semantic analysis, coupled with condition/conclusion/exception analysis on the query and relevant articles. After constructing a set of pre-trained semantic word embeddings using *word2vec*[2], we train the model to learn models for semantic matching between question and corresponding articles. These feature extraction methods are coupled with negation analysis, then used to construct an SVM model to provide the required yes/no answers.

2. Phase 1: Legal Information Retrieval

2.1 IR Models

[1] https://webdocs.cs.ualberta.ca/~miyoung2/COLIEE2016/

[2] code.google.com/p/word2vec

[3] Lucene can be downloaded from http://lucene.apache.org/core/.

Our information retrieval model is a combination of the term frequency–inverse document frequency (tf-idf) model and a support vector machine (SVM) re-ranking model. We will describe the two components in the following.

2.1.1 the tf-idf model

One of our baseline models is a tf-idf model implemented in Lucene, an open source IR system[3].

The simplified version of Lucene's similarity score of an article to a query is:

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

The score *tf-idf*(*Q,A*) is a measure which computes 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*. The final score is the sum of the scores of terms in both the article and the query. The bigger *tf-idf*(*Q,A*), the more relevance between the query *Q* and the article *A*.

The choice of terms in documents is as important as choosing the score functions. Instead of using the original words in a text, we lemmatize the text with the Stanford NLP tool [15]. After lemmatization, words such as *steal, stole, and steals* become *steal*. In this way, if there is *steal* in the question, but *stole* in the article, we can still retrieve the article as a match.

2.1.2 The ranking SVM

Previous tf-idf models rank the articles based on frequency information. However, other features, such as the matched phrases between the article and the queries, are useful too. We use an SVM Ranking model to learn the importance of such features and then re-estimate the score of each retrieved article from the tf-idf output.

The ranking SVM model was proposed by [2]. The model ranks the documents based on user's click through data, in our case, the correct articles in the training data. Given the articles retrieved from the tf-idf model, ranking SVM will learn to rank correct articles higher than incorrect ones. More precisely, given the feature vector of a training instance, i.e. a retrieved article set given a query, denoted by $\Phi(Q,A_i)$, the model tries to find a ranking that satisfies constraints:

$$\Phi(Q, A_i) > \Phi(Q, A_j) \quad (2)$$

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

To use this ranking SVM, we incorporate the following types of features:

- 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

[3] Lucene can be downloaded from http://lucene.apache.org/core/.

instance representation, this type is converted into binary features whose values are one or zero, i.e. if a word exists in the intersection word set or not.

- Dependency pairs: word pairs that are linked by a dependency link, arising from a dependency parsing. 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).

2.2 Experiments

The COLIEE legal IR task has several sets of queries with the Japan civil law articles as documents (1044 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 8 sets of queries for a dry run. The 8 sets of data include 412 queries. We used the corresponding 8-fold leave-one-out cross validation evaluation. The metric for measuring our IR models is Mean Average Precision (MAP):

$$\mathrm{MAP}(Q) = \frac{1}{|Q|}\sum_{q\in Q}\frac{1}{m}\sum_{k\in(1,m)}\mathrm{precision}(R_k) \tag{3}$$

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 first until the *k-th* article. In the following experiments, we set m as 3 for all queries, corresponding to the column MAP@3 in Table 1. SVM's parameters are set according to the 8-fold cross validation IR performance. Given the top 20 articles returned by the tf-idf model, the SVM model extracts features for every article and trains according to the order that correct articles are ranked higher than incorrect ones. Table 1 presents the results of different models. The result shows that the ranking SVM with all three features achieves the highest performance. We also show the standard deviation of the cross-validation, the smallest and largest MAP@3 for 8 folds to show the effect of small training data. It seems the TF-IDF model causes larger deviation than the SVM models.

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

Id	models	MAP@3	Standard deviation	Smallest	Largest
1	tf-idf with lemma	39.8	7.0	23.8	45.5
2	SVM-ranking with lemma	39.8	6.5	26.1	46.8
3	SVM-ranking with lemma and dependency pair	41.2	5.4	30.1	48.4
4	model 3 plus tf-idf score	43.1	7.1	27.2	49.0

3. Phase 2: Answering yes/no Questions

Our system uses word embedding for semantic analysis and paraphrasing for term expansion to predict textual entailment. We will look at the entailment types and extract features from sentences, as described in detail in the next subsections.

3.1 Entailment types

We identify a variety of types of entailment as shown in Table 2. By classifying a yes/no problem as one of these types, we can determine what kind of further information is required to provide a decision on entailment.

Table 2. Query-Article types

Query-Article type	Proportion	Query-Article type	Proportion
One article refers to another article	0.182	Question is a specific example	0.092
Multiple relevant articles	0.388	Multiple conditions	0.731
Exceptional case	0.148		

Table 2 shows our list of query-article types. Note that one article can refer to another, such as "*If there is any latent defect in the subject matter of a sale, the provisions of Article 566 shall apply mutatis mutandis."* This makes textual entailment more complex as we also need to analyze the meaning of the referred article.

In another case, one query can have multiple relevant articles, so we have to combine the multiple articles' meanings or choose one as most relevant for determining entailment.

Note also that many statutes have exceptional cases, so we need to recognize if the query is included in the exceptional case or not. In addition, a query may be one example of the article case. There are also cases where some articles have multiple conditions for one conclusion, so we must then confirm if each condition is satisfied in the query. Overall, many query-article types also require the identification of negation and synonym/antonym relations to confirm the correct entailment.

The overall description of our procedure of textual entailment is as follows:

1. Find the most relevant article with a query
2. Divide a query and the corresponding article into "Condition(s)," "Conclusion," and "Exception-condition(s)."
3. Term expansion using Paraphrasing
4. Negation and synonym detection
5. Extract features and perform learning using the features

In the following subsections, we explain each step in detail.

3.2 Finding the most relevant article/sentences

In case that there are multiple relevant sentences, we choose the article with the most overlapping words with the query. In the chosen article, if there exist multiple regulations, we also choose the one regulation that has most overlapping words with a query.

3.3 Negation and synonym detection

We exploit a process for managing negation and antonyms as described in Kim et al. [10]. In addition, we approximate word semantic similarity by converting words to vector representations using the *word2vec* tool. The output of the semantic similarity is vector similarity. We used 1,044 legal law articles to train the *word2vec* tool (*vector dimension*=50).

3.4 Condition/Conclusion/Exception detection

From our analysis of the structure of statutes we extract components based on the following rules:

$$conclusion := segment_{last}(sentence, keyword),$$

$$condition := \sum_{i \neq last} segment_i(sentence, keyword),$$

$$condition := condition \text{ [or] } condition$$

$$condition := sub\_condition \text{ [and] } sub\_condition$$

$$exception\_conclusion := segment_{last}(sentence, exception\_keyword),$$

$$exception\_condition := \sum_{i \neq last} segment_i(sentence, exception\_keyword),$$

$$exception\_condition := exception\_condition \text{ [or] } exception\_condition$$

$$exception\_condition := sub\_exception\_condition \text{ [and] } sub\_exception\_condition$$

So from keywords of a condition, we segment sentences. The keywords of the condition are as follows: “in case(s),” “if,” “unless,” “with respect to,” “when,” and “, (comma).” After this segmentation, the last segment is considered to be a conclusion, and the rest of the sentence is considered as a condition. (We used the symbol ∑ to denote the concatenation of the segments). We also distinguish segments which denote exceptional cases. Currently, we take the *exception_keyword* indication as “... this shall not apply, if (unless).”

The original bar law examinations in the COLIEE data are provided in Japanese and English, and our initial implementation used a Korean translation, provided by the Excite translation tool.[4] 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. Therefore, the above rules may not be appropriate for all English sentences, because the segment order can differ.

The following is an example of condition and conclusion detection:

> <Civil law example>A person who employs others for a certain business, shall be liable for damages inflicted on a third party by his/her employees with respect to the execution of that business; Provided, however, that this shall not apply, if the employer exercised reasonable care in appointing the employee or in supervising the business, or if the damages could not have been avoided even if he/she had exercised reasonable care.
>
> *(1) Conclusion* => shall be liable for damages inflicted on a third party by his/her employees with respect to the execution of that business.
> *(2) Condition* => A person who employs others for a certain business
> *(3) Exception*
> *Conclusion* => this shall not apply (opposite of main conclusion)
> *Condition*
> *Condition* =>
> *Condition* => if the employer exercised reasonable care in appointing the employee
> *Condition (OR)* => in supervising the business

[4] http://excite.translation.jp/world/

Condition(OR) => if the damages could not have been avoided even if he/she had exercised reasonable care.

3.5 Term expansion using paraphrasing

There are many words with similar meanings but different lexical forms (e.g., 'obligor' vs. 'debtor', 'rescind' vs. 'cancel', 'lien' vs. 'privilege', etc.). To resolve these diverse terms, we use language translation-based paraphrasing. The idea of translation-based paraphrase is that translating from one language to another and then back, will often produce semantically similar but lexically distinct outputs. If we assume that the language translations preserve semantics, more or less, then lexically distinct terms can be considered as paraphrases. In our application of this idea, we translate the original English query/document into German, and then back-translate the German sentences into English. We then can detect pairs of words/phrases which can be considered as semantically related: the original English sentence and double-translated English sentence. We used Google translate, and chose German as the pivot language, which is a closely related to English, which we hope reduces the translation errors.

We performed double translation with 100 article laws in the Japanese Civil Code. We used the monolingual alignment tool of Sultan et al. [12] to create automatic word alignments in English. Table 3 shows examples of detected paraphrases using language translation. We can see that it also detects plural forms and past tense forms, in addition to words with similar meanings. We extract the top 100 paraphrases, and manually extracted corresponding Korean words in the Korean-translated Query-Article text.

Table 3 Examples of Detected Paraphrases

Original word	Paraphrased word	Original word	Paraphrased word
year	years	establishes	sets
makes	made	purpose	aim
warranties	guarantees	matter	area
released	relieved	pledge	commitment
assigned	transferred	demand	claim
respect	relation	referred	designated

3.6 Supervised learning with SVM

Since we cannot anticipate the impact of each linguistic attribute, we run a machine learning algorithm that learns what information is relevant in the text to achieve our goal. We have compared our method with SVM, as a kind of supervised learning model. Using the SVM tool included in the Weka [4] software library, we performed cross-validation for the 412 questions. We used a linear kernel SVM because it is popular for real-time applications as they enjoy both faster training and classification speeds (significantly reduced memory requirements compared with non-

linear kernels, because of the compact representation of the decision function). We used the following features:

a) Word Lemma
b) Lexical semantic features
c) Negation feature
d) Sentence analysis feature (condition, conclusion, and exception)

For concept features, we have exploited word embedding using *word2vec*. When we use word embedding, we assume the concepts of two words are the same if their vector cosine similarity >=0.8.

The detailed features that we use are as follows:

Feature 1: $If\ \exists_{i,j}\{(\text{concept}(w_i)\ ,\ \text{Query}_{\text{condition}}) \cap (\text{concept}(w_j)\ ,\ \text{Article}_{\text{condition}})\}$

Feature 2: $If\ \exists_{i,j}\{(\text{concept}(w_i)\ ,\ \text{Query}_{\text{conclusion}}) \cap (\text{concept}(w_j)\ ,\ \text{Article}_{\text{conclusion}})\}$

Feature 3: $If\ \exists Article_{sub\_condition} \cap$
$\exists_{i,j,k}\{(\text{concept}(w_i)\ ,\ \text{Query}_{\text{condition}}) \cap (\text{concept}(w_j)\ ,\ \text{Article}_{\text{sub\_condition}_k})=\varnothing\}$

Feature 4: $If\ \exists_{i,j}\{(\text{concept}(w_i)\ ,\ \text{Query}_{\text{condition}}) \cap (\text{concept}(w_j)\ ,\ \text{Article}_{\text{exception\_condition}})\}$

Feature 5: $If\ \exists Article_{sub\_exception\_condition} \cap$
$\exists_{i,j,k}\{(\text{concept}(w_i)\ ,\ \text{Query}_{\text{condition}}) \cap (\text{concept}(w_j)\ ,\ \text{Article}_{\text{sub\_exception\_condition}_k})=\varnothing\}$

Feature 6 : $If\ neg\_level(\text{Query}_{\text{condition}}) = neg\_level(\text{Article}_{\text{condition}})$

Feature 7 : $If\ neg\_level(\text{Query}_{\text{conclusion}}) = neg\_level(\text{Article}_{\text{conclusion}})$

Feature 8 : $If\ neg\_level(\text{Query}_{\text{condition}}) = neg\_level(\text{Article}_{\text{exception\_condition}})$

Features 1 and 2 check if there are overlapped concepts between a query condition (conclusion) and its relevant article condition (conclusion). Feature 3 checks if there is overlapped word between a query condition and its relevant article sub-condition. Because the article sub-condition is connected with other sub-condition(s), using "and" as a connector, the query should include the meanings of all the article sub-conditions. Feature 4 checks if there are overlapped concepts between a query condition and its article exception-condition. We want to check if the query is included in the exceptional case using the feature. Feature 5 checks if there is no overlapped word between a query condition and its relevant article sub-exception-condition. Features 6, 7, and 8 check the negation levels between the query condition, article condition, query conclusion, article conclusion, and article exception-condition. We compute negation value (neg-level()) according to Kim et al. [10].

4. Phase 2: Experimental Results

4.1 Experimental setup for Phase 2

Table 4. Experimental results on dry run data for Phase 2

Method	Accu(%)
(a) Baseline	51.70
(b) Our method using cross-validation with Supervised learning (SVM) not using word embedding but using lexical word itself	62.14
(c) Our method using cross-validation with Supervised learning (SVM) using word embedding	60.67
(d) Cross-validation using Kim et al. [10]	60.92
(e) Cross-validation using Kim et al. [11]	61.65
Without term expansion using paraphrasing from (b)	60.44
Without neg_level() from (b)	49.27

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 the questions are entailed by the corresponding civil law articles.

There is a balanced positive-negative sample distribution in the dataset (51.70% yes, and 48.30% no) for a dry run of COLIEE 2016 dataset, so we consider the baseline for true/false evaluation is the accuracy when always returning "yes," which is 51.70%. Our total data for the dry run has 412 questions.

4.2 Experimental results

Table 4 shows the experimental results. An SVM-based model showed accuracy of 62.14% when we did not use word embedding but used the lexical form of each word; the method of Kim et al. [10] showed 60.92% and that of Kim et al. [11] showed 61.65%. Our SVM augmented system outperformed Kim et al. [10, 11].

Table 4 also shows the experimental results arising from changing some features in our method. The accuracy was lower by 1.70% when we removed paraphrasing, and the accuracy was lower by 1.47% when we used word embedding, which suggests that word embedding does not help capture the semantics better than the lexical word itself. When we did not use the negation feature, the accuracy became lower by 12.87%, which proves that the negation feature is most valuable.

4.3 Error analysis

From unsuccessful instances, we manually classified the error types as shown in Table 5. The biggest error arises, of course, from the semantic similarity error, which we believe our word embedding is not enough to capture the semantic similarity. We will try to include the bar exam text in the training data for the word embedding. 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. We also found cases that indicate the need to do more extensive temporal analysis.

Table 5. Error types

Error type	Accuracy (%)	Error type	Accu.(%)
Specific example case	9.62	Semantic similarity error	28.85
Incorrect detection of the most similar article sentence	10.90	Constraints in condition	25.00
Incorrect detection of Condition, Conclusion, and mismatch	11.54	Etc.	14.10

It will be interesting if we compare our performance using Korean-translated sentences with that using original Japanese sentences. We would expect the system using original sentences will show improved performance, because there exist no translation errors. As future work, we will construct a Japanese system using paraphrase/synonym/antonym dictionaries for Japanese, and then analyze how the translation affects performance.

5 Related work

A previous textual entailment method from W. Bdour et al. [5] provided the basis for a yes/no Arabic question answering system. They used a kind of logical representation, and compared the logical representation between queries and documents. This method will be appropriate for the task where queries and documents have similar logical representations so it is easier to confirm entailment from one logical representation to another. However, our task's entailment type is more complex, so we take an approach that approximates the logical content of queries and documents, rather than attempt any complete transformation to a logical form.

Nielsen et al. [6] extracted features from dependency paths, and combined them with word-alignment features in a mixture of an expert-based classifier. Zanzotto et al. [7] proposed a syntactic cross-pair similarity measure for RTE. Harmeling [8] took a

similar classification-based approach with transformation sequence features. Marsi et al. [9] described a system using dependency-based paraphrasing techniques.

Many methods have been proposed for paraphrasing. One of the methods is the idea of semantic parsing via paraphrasing [13]. They transform a sentence into a logical form, and then convert logical forms to canonical form using the Freebase database. Subsequently, they obtain an association between the original sentence and canonical forms. However, hundreds of logical/canonical forms have been generated per sentence in their method, and the method does not show how to choose the best amongst them.

The method of Zhang et al. [14] also uses a pivot language for paraphrasing. Like us, they translate one language to another, then re-translate from the translated language into the original language. They then obtain a paraphrasing set between the original utterance and double-translated utterance. They showed improved performance in paraphrase detection using the pivot language translation, so we also employ the language translation-based paraphrasing. But instead of their use of the GIZA++ alignment, we used the monolingual alignment tool of Sultan et al. [12], because GIZA++, which is for alignment between two different languages, did not show good performance for our dataset.

6 Conclusion

We have described our most recent implementation for the Competition on Legal Information Extraction/Entailment (COLIEE)-2016 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 SVM model using paraphrasing and pre-trained word embedding and query/article condition/conclusion/exception analysis. We show improved performance over a previous system, and paraphrasing and negation detection contributed to the performance.

Acknowledgements

This research was supported by the Alberta Innovates Centre for Machine Learning (AICML). 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 specicity 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 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
6. 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
7. 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
8. 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
9. 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
10. M-Y. Kim, Y. Xu, and R. Goebel. Alberta-KXG: Legal Question Answering Using Ranking SVM and Syntactic/Semantic Similarity. JURISIN Workshop, 2014
11. M-Y. Kim, Y. Xu, and R. Goebel. A Convolutional Neural Network in Legal Question Answering. JURISIN Workshop 2015.
11. M. A. Sultan, S. Bethard, and T, Sumner. Back to basics for monolingual alignment: Exploiting word similarity and contextual evidence. Transactions of the Association for Computational Linguistics, 2, pp. 219-230, 2014
12. J. Berant, and L. Percy. Semantic Parsing via Paraphrasing, Proceedings of the conference of the Association for Computational Linguistics (ACL), pp. 1415-1425, 2014
13. W. Zhang, Z. Ming, Y. Zhang, T. Liu and T. S. Chua. Exploring Key Concept Paraphrasing Based on Pivot Language Translation for Question Retrieval. In AAAI, pp. 410-416, 2015
14. Christopher D. Manning, M. Surdeanu, J. Bauer, J. Finkel, S. J. Bethard, and D. McClosky. *The Stanford CoreNLP Natural Language Processing Toolkit*. In Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations, pp. 55-60, 2014