Legal Question Answering using Ranking SVM and Deep Convolutional Neural Network

Abstract

This paper presents a study of employing Ranking SVM and Convolutional Neural Network for two missions: legal information retrieval and question answering in the Competition on Legal Information Extraction/Entailment. For the first task, our proposed model used a triple of features (LSI, Manhattan, Jaccard), and is based on paragraph level instead of article level as in previous studies. In fact, each single-paragraph article corresponds to a particular paragraph in a huge multiple-paragraph article. For the legal question answering task, additional statistical features from information retrieval task integrated into Convolutional Neural Network contribute to higher accuracy.

Introduction

Legal text, along with other natural language text data, e.g. scientific literature, news articles or social media, has seen an exponential growth on the Internet and in specialized systems. Unlike other textual data, legal texts contain strict logical connections of law-specific words, phrases, issues, concepts and factors between sentences or various articles. Those are for helping people to make a correct argumentation and avoid ambiguity when using them in a particular case. Unfortunately, this also makes information retrieval and question answering on legal domain become more complicated than others.

There are two primary approaches to information retrieval (IR) in the legal domain BIBREF0 : manual knowledge engineering (KE) and natural language processing (NLP). In the KE approach, an effort is put into translating the way legal experts remember and classify cases into data structures and algorithms, which will be used for information retrieval. Although this approach often yields a good result, it is hard to be applied in practice because of time and financial cost when building the knowledge base. In contrast, NLP-based IR systems are more practical as they are designed to quickly process terabytes of data by utilizing NLP techniques. However, several challenges are presented when designing such system. For example, factors and concepts in legal language are applied in a different way from common usage BIBREF1 . Hence, in order to effectively answer a legal question, it must compare the semantic connections between the question and sentences in relevant articles found in advance BIBREF2 .

Given a legal question, retrieving relevant legal articles and deciding whether the content of a relevant article can be used to answer the question are two vital steps in building a legal question answering system. Kim et al. BIBREF2 exploited Ranking SVM with a set of features for legal IR and Convolutional Neural Network (CNN) BIBREF3 combining with linguistic features for question answering (QA) task. However, generating linguistic features is a non-trivial task in the legal domain. Carvalho et al. BIBREF1 utilized n-gram features to rank articles by using an extension of TF-IDF. For QA task, the authors adopted AdaBoost BIBREF4 with a set of similarity features between a query and an article pair BIBREF5 to classify a query-article pair into “YES" or “NO". However, overfitting in training may be a limitation of this method. Sushimita et al. BIBREF6 used the voting of Hiemstra, BM25 and PL2F for IR task. Meanwhile, Tran et al. BIBREF7 used Hidden Markov model (HMM) as a generative query model for legal IR task. Kano BIBREF8 addressed legal IR task by using a keyword-based method in which the score of each keyword was computed from a query and its relevant articles using inverse frequency. After calculating, relevant articles were retrieved based on three ranked scores. These methods, however, lack the analysis of feature contribution, which can reveal the relation between legal and NLP domain. This paper makes the following contributions:

In the following sections, we first show our idea along with data analysis in the context of COLIEE. Next, we describe our method for legal IR and legal QA tasks. After building a legal QA system, we show experimental results along with discussion and analysis. We finish by drawing some important conclusions.

Basic Idea

In the context of COLIEE 2016, our approach is to build a pipeline framework which addresses two important tasks: IR and QA. In Figure 1 , in training phase, a legal text corpus was built based on all articles. Each training query-article pair for LIR task and LQA task was represented as a feature vector. Those feature vectors were utilized to train a learning-to-rank (L2R) model (Ranking SVM) for IR and a classifier (CNN) for QA. The red arrows mean that those steps were prepared in advance. In the testing phase, given a query q , the system extracts its features and computes the relevance score corresponding to each article by using the L2R model. Higher score yielded by SVM-Rank means the article is more relevant. As shown in Figure 1 , the article ranked first with the highest score, i.e. 2.6, followed by other lower score articles. After retrieving a set of relevant articles, CNN model was employed to determine the “YES" or “NO" answer of the query based on these relevant articles.

Data Observation

The published training dataset in COLIEE 2016 consists of a text file containing Japanese Civil Code and eight XML files. Each XML file contains multiple pairs of queries and their relevant articles, and each pair has a label “YES" or “NO", which confirms the query corresponding to the relevant articles. There is a total of 412 pairs in eight XML files and 1,105 articles in the Japanese Civil Code file, and each query can have more than one relevant articles.

After analyzing the dataset in the Civil Code file, we observed that the content of a query is often more or less related to only a paragraph of an article instead of the entire content. Based on that, each article was treated as one of two types: single-paragraph or multiple-paragraph, in which a multiple-paragraph article is an article which consists of more than one paragraphs. There are 7 empty articles, 682 single-paragraph articles and the rest are multiple-paragraph.

Based on our findings, we proposed to split each multiple-paragraph article into several independent articles according to their paragraphs. For instance, in Table 1 , the Article 233 consisting of two paragraphs was split into two single-paragraph articles 233(1) and 233(2). After splitting, there are in total 1,663 single-paragraph articles.

Stopwords were also removed before building the corpus. Text was processed in the following order: tokenization, POS tagging, lemmatization, and stopword removal. In BIBREF1 , the stopword removal stage was done before the lemmatization stage, but we found that after lemmatizing, some words might become stopwords, for instance, “done" becomes “do". Therefore, the extracted features based on words are more prone to be distorted, leading to lower ranking performance if stopword removal is carried out before lemmatization step. Terms were tokenized and lemmatized using NLTK, and POS tagged by Stanford Tagger.

Legal Information Retrieval

In order to build a legal IR, traditional models such as TF-IDF, BM25 or PL2F can be used to generate basic features for matching documents with a query. Nevertheless, to improve not only the accuracy but also the robustness of ranking function, it is essential to take into account a combination of fundamental features and other potential features. Hence, the idea is to build a L2R model, which incorporates various features to generate an optimal ranking function.

Among different L2R methods, Ranking SVM (SVM-Rank) BIBREF9 , a state-of-the-art pairwise ranking method and also a strong method for IR BIBREF10 , BIBREF11 , was used. Our model is an extended version of Kim's model BIBREF2 with two new aspects. Firstly, there is a big distinction between our features and Kim's features. While Kim used three types of features: lexical words, dependency pairs, and TF-IDF score; we conducted a series of experiments to discover a set of best features among six features as shown in Table 2 . Secondly, our model is applied to individual paragraphs as described in section "Data Observation" instead of the whole articles as in Kim's work.

Given n training queries $\lbrace q_i\rbrace _{i=1}^{n}$, their associated document pairs $(x_u^{(i)},x_v^{(i)})$ and the corresponding ground truth label $y_{u,v}^{(i)}$, SVM Rank optimizes the objective function shown in Equation (13) subject to constraints (14), and (15):

$$min \quad \frac{1}{2}\Vert w\Vert ^2 + \lambda \sum _{i=1}^{n}\sum _{u,v:y_{u,v}^{(i)}} \xi _{u,v}^{(i)}$$ (Eq. 13)

$$s.t. \quad w^T(x_u^{(i)} - x_v^{(i)}) \ge 1 - \xi _{u,v}^{(i)} \quad \text{if} \quad y_{u,v}^{(i)}=1$$ (Eq. 14)

where: $f(x)=w^Tx$ is a linear scoring function, (x_u,x_v) is a pairwise and $\xi _{u,v}^{(i)}$ is the loss. The document pairwise in our model is a pair of a query and an article.

Based on the corpus constructed from all of the single-paragraph articles (see Section "Data Observation"), three basic models were built: TF-IDF, LSI and Latent Dirichlet Allocation (LDA) BIBREF12 . Note that, LSI and LDA model transform articles and queries from their TF-IDF-weighted space into a latent space of a lower dimension. For COLIEE 2016 corpora, the dimension of both LSI and LDA is 300 instead of over 2,100 of TF-IDF model. Those features were extracted by using gensim library BIBREF13 . Additionally, to capture the similarity between a query and an article, we investigated other potential features described in Table 2 . Normally, the Jaccard coefficient measures similarity between two finite sets based on the ratio between the size of the intersection and the size of the union of those sets. However, in this paper, we calculated Generalized Jaccard similarity as:

$$ J(q,A) = J(X,Y) = \frac{\sum _{i}^{} min(x_i,y_i)}{\sum _{i}^{} max(x_i,y_i)}$$ (Eq. 16)

and Jaccard distance as:

$$ D(q,A) = 1 - J(q,A)$$ (Eq. 17)

where $X = \lbrace x_1,x_2,..,x_n\rbrace $ and $Y=\lbrace y_1,y_2,...,y_n\rbrace $ are two TF-IDF vectors of a query q and an article A respectively.

The observation in Section "Data Observation" also indicates that one of the important properties of legal documents is the reference or citation among articles. In other words, an article could refer to the whole other articles or to their paragraphs. In BIBREF1 , if an article has a reference to other articles, the authors expanded it with words of referential ones. In our experiment, however, we found that this approach makes the system confused to rank articles and leads to worse performance. Because of that, we ignored the reference and only took into account individual articles themselves. The results of splitting and non-splitting are shown in Table 5 .

Legal Question Answering

Legal Question Answering is a form of textual entailment problem BIBREF14 , which can be viewed as a binary classification task. To capture the relation between a question and an article, a set of features can be used. In the COLLIE 2015, Kim BIBREF3 efficiently applied Convolution Neural Network (CNN) for the legal QA task. However, the small dataset is a limit of deep learning models. Therefores, we provided additional features to the CNN model.

The idea behind the QA is that we use CNN BIBREF2 with additional features. This is because: (i) CNN is capable to capture local relationship between neighboring words, which helps CNN to achieve excellent performance in NLP problems BIBREF15 , BIBREF2 , BIBREF16 , BIBREF17 and (ii) we can integrate our knowledge in legal domain in the form of statistical features, e.g. TF-IDF and LSI.

In Figure 2 , the input features $v_1,v_2,...,v_{400}$ are constructed and fed to the network as follows :

$v_1,v_3,v_5,...,v_{399}$: a word embedding vector of the question sentence

$v_2,v_4,...,v_{400}$: a word embedding vector of the most relevant article sentence

A sentence represented by a set of words was converted to a word embedding vector v_1^{200} by using bag-of-words model (BOW) BIBREF18 . BOW model generates a vector representation for a sentence by taking a summation over embedding of words in the sentence. The vector is then normalized by the length of the sentence:

$$s= \frac{1}{n}\sum _{i= 1}^{n}s_{i}$$ (Eq. 22)

where: s is a d -dimensional vector of a sentence, s_{i} is a d -dimensional vector of i^{th} word in the sentence, n is the length of sentence. A word embedding model ($d=200$) was trained by using Word2Vec BIBREF19 on the data of Japanese law corpus BIBREF1 . The corpus contains all Civil law articles of Japan's constitution with 13.5 million words from 642 cleaned and tokenized articles.

A filter was denoted as a weight vector w with length h ; w will have h parameters to be estimated. For each input vector $S \in \mathbb {R}^{d} $, the feature map vector $O \in \mathbb {R}^{d-h+1}$ of the convolution operator with a filter w was obtained by applying repeatedly w to sub-vectors of S :

$$o_{i}=w\cdot S[i:i+h-1]$$ (Eq. 24)

where: $i=0,1,2,...,d-h+1$ and ($\cdot $) is dot product operation.

Each feature map was fed to a pooling layer to generate potential features by using the average mechanism BIBREF20 . These features were concatenated to a single vector for classification by using Multi-Layer Perceptron with sigmoid activation. During training process, parameters of filters and perceptrons are learned to optimize the objective function.

In our model, 10 convolution filters (length = 2) were applied to two adjacent input nodes because these nodes are the same feature type. An average pooling layer (length = 100) is then utilized to synthesize important features. To enhance the performance of CNN, two additional statistic features: TF-IDF and LSI were concatenated with the result of the pooling layer, then fed them into a 2-layer Perceptron model to predict the answer.

In Legal QA task, the proposed model was compared to the original CNN model and separate TF-IDF, LSI features. For evaluation, we took out 10% samples from training set for validation, and carried out experiments on dataset with balanced label distribution for training set, validation set and testing set.

In CNN models, we found that these models are sensitive to the initial value of parameters. Different values lead to large difference in results ($\pm $ 5%). Therefore, each model was run n times (n=10) and we chose the best-optimized parameters against the validation set. Table 7 shows that CNN with additional features performs better. Also, CNN with LSI produces a better result as opposed to CNN with TF-IDF. We suspect that this is because TF-IDF vector is large but quite sparse (most values are zero), therefore it increases the number of parameters in CNN and consequently makes the model to be overfitted easily.

To achieve the best configuration of CNN architecture, the original CNN model was run with different settings of number filter and hidden layer dimension. According to Table 8 , the change of hyperparameter does not significantly affect to the performance of CNN. We, therefore, chose the configuration with the best performance and least number of parameters: 10 filters and 200 hidden layer size.

Information Retrieval

For information retrieval task, 20% of query-article pairs are used for evaluating our model while the rest is for training. As we only consider single-paragraph articles in the training phase, if a multiple-paragraph article is relevant, all of its generated single-paragraph articles will be marked as relevant. In addition, the label for each query-article pair is set either 1 (relevant) or 0 (irrelevant). In our experiment, instead of selecting top k retrieved articles as relevant articles, we consider a retrieved article A_i as a relevant article if its score S_i satisfies Equation (26):

$$\frac{S_i}{S_0} \ge 0.85$$ (Eq. 26)

where: S_0 is the highest relevant score. In other words, the score ratio of a relevant article and the most relevant article should not be lower than 85% (choosing the value 0.85 for this threshold is simply heuristic based). This is to prevent a relevant article to have a very low score as opposed to the most relevant article.

We ran SVM-Rank with different combinations of features listed in Table 2 , but due to limited space, we only report the result of those combinations which achieved highest F1-score. We compared our method to two baseline models TF-IDF and LSI which only use Cosine similarity to retrieve the relevant articles. Results from Table 3 indicate that (LSI, Manhattan, Jaccard) is the triple of features which achieves the best result and the most stability.

The contribution of each feature was investigated by using leave-one-out test. Table 4 shows that when all six features are utilized, the F1-score is approximately 0.55. However when excluding Jaccard, F1-score drops to around 0.5. In contrast, when other features are excluded individually from the feature set, the result remains stable or goes up slightly. From this result, we conclude that Jaccard feature significantly contributes to SVM-Rank performance.

We also analyzed the contribution of feature groups to the performance of SVM-Rank. When removing different triples of features from the feature set, it can be seen that (TF-IDF, Manhattan, Jaccard) combination witnesses the highest loss. Nevertheless, as shown in Table 3 , the result of (LSI, Manhattan, Jaccard) combination is more stable and better.

As mentioned, we proposed to split a multiple-paragraph article into several single-paragraph articles. Table 5 shows that after splitting, the F1-score performance increases by 0.05 and 0.04 with references and without references respectively. In both cases (with and without the reference), using single-paragraph articles always results a higher performance.

Results from Table 5 also indicate that expanding the reference of an article negatively affects the performance of our model, reducing the F1-score by more than 0.02. This is because if we only expand the content of an article with the content of referential one, it is more likely to be noisy and distorted, leading to lower performance. Therefore, we conclude that a simple expansion of articles via their references does not always positively contribute to the performance of the model.

Since linear kernel was used to train the SVM-Rank model, the role of trade-off training parameter was analyzed by tuning C value from 100 to 2000 with step size 100. Empirically, F1-score peaks at 0.6087 with C = 600 when it comes to COLIEE 2016 training dataset. We, therefore, use this value for training the L2R model.

Formal run phase 1 - COLIEE 2016

In COLIEE 2016 competition, Table 6 shows the top three systems and the baseline for the formal run in phase 1 BIBREF21 . Among 7 submissions, iLis7 BIBREF22 was ranked first with outstanding performance (0.6261) by exploiting ensemble methods for legal IR. Several features such as syntactic similarity, lexical similarity, semantic similarity, were used as features for two ensemble methods Least Square Method (LSM) and Linear Discriminant Analysis (LDA).

HUKB-2 BIBREF23 used a fundamental feature BM25 and applied mutatis mutandis for articles. If both an article and a query have conditional parts, they are divided into two parts like conditional parts and the rest part before measuring their similarity. This investigation in conditional parts is valuable since it is a common structure in laws. Their F1-score in formal rune is the second highest (0.5532), which is slightly higher than our system (0.5478) using SVM-Rank and a set of features LSI, Manhattan, Jaccard. This shows that for phase 1, our model with a set of defined features is relatively competitive.

Legal Question Answering System

In this stage, we illustrate our framework on COLIEE 2016 data. The framework was trained on XML files, from H18 to H23 and tested on XML file H24. Given a legal question, the framework first retrieves top five relevant articles and then transfers the question and relevant articles to CNN classifier. The running of framework was evaluated with 3 scenarios:

No voting: taking only a top relevant article to use for predicting an answer for that question.

Voting without ratio: each of results, which is generated by applying our Textual entailment model to each article, gives one vote to the answer which it belongs to. The final result is the answer with more votes.

Voting with ratio: similar to Voting without ratio. However, each of results gives one vote corresponding to article's relevant score. The final result is the answer with higher voting score.

Table 9 shows results with different scenarios. The result of No voting approach is influenced by IR task's performance, so the accuracy is not as high as using voting. The relevant score disparity between the first and second relevant article is large, which causes a worse result of Voting with ratio compared to Voting without ratio.

Formal run phase 2 & 3 - COLIEE 2016

Table 10 lists the state-of-the art methods for the formal run 2016 in phase 2 and 3. In phase 2, two best systems are iList7 and KIS-1. iList7 applies major voting of decision tree, SVM and CNN with various features; KIS-1 just uses simple rules of subjective cases and an end-of-sentence expression. In phase 3, UofA achives the best score. It extracts the article segment which related to the query. This system also performs paraphrasing and detects condition-conclusion-exceptions for the query/article. From the experimental results, deep learning models do not show their advantages in case of a small dataset. On the other hand, providing handcraft features and rules are shown to be useful in this case.

Splitting and non-splitting error analysis

In this section, we show an example in which our proposed model using single-paragraph articles gives a correct answer in contrast with utilizing non-splitting one. Given a query with id H20-26-3: “A mandate contract is gratuitous contract in principle, but if there is a special provision, the mandatary may demand renumeration from the mandator.”, which refers to Article 648:

Apparently, three paragraphs and the query share several words namely mandatary, remuneration, etc.

In this case, however, the correct answer is only located in paragraph 1, which is ranked first in the single-paragraph model in contrast to two remaining paragraphs with lower ranks, 5th and 29th as shown in Table 12 .

Interestingly, Article 653 has the highest relevant score in non-splitting method and rank 2nd in splitting approach. The reason for this is that Article 653 shares other words like mandatary, mandator as well. Therefore, it makes retrieval system confuse and yield incorrect order rank. By using single-paragraph, the system can find more accurately which part of the multiple-paragraph article is associated with the query's content.

Conclusion

This work investigates Ranking SVM model and CNN for building a legal question answering system for Japan Civil Code. Experimental results show that feature selection affects significantly to the performance of SVM-Rank, in which a set of features consisting of (LSI, Manhattan, Jaccard) gives promising results for information retrieval task. For question answering task, the CNN model is sensitive to initial values of parameters and exerts higher accuracy when adding auxiliary features.

In our current work, we have not yet fully explored the characteristics of legal texts in order to utilize these features for building legal QA system. Properties such as references between articles or structured relations in legal sentences should be investigated more deeply. In addition, there should be more evaluation of SVM-Rank and other L2R methods to observe how they perform on this legal data using the same feature set. These are left as our future work.

Acknowledgement

This work was supported by JSPS KAKENHI Grant number 15K16048, JSPS KAKENHI Grant Number JP15K12094, and CREST, JST.