THUIR@AILA 2019: Information Retrieval Approaches for Identifying Relevant Precedents and Statutes

Yunqiu Shao[1] and Ziyi Ye[1]

BNRist, DCST, Tsinghua University, Beijing 100084, China
`shaoyq18, zy-ye16@mails.tsinghua.edu.cn`

1

Abstract. In this paper, we present the methodologies employed in the AILA 2019. There are two tasks in this challenge. Given a situation, the first task is to identify relevant precedents, and the second one is to identify relevant statutes. We consider both tasks as ranking problems, and combine ranking models with auto summarization and data processing techniques in generating our runs. We participated in both tasks, and our three runs achieved the best performance in Task 2.

Keywords: Information Retrieval · Precedent · Statute.

1 Introduction

Precedents and statutes are both primary sources in a Common Law System. With the rapid increase of digitized legal documents, it is important to build an automatic legal retrieval system which can identify relevant precedents and statutes. In recent years, there have been some works on the topic of legal information retrieval. The FIRE 2017 IRLeD Track [3] focused on creating a framework for legal keywords extraction and precedent retrieval. COLIEE has been held annually to develop techniques of retrieval and entailment in the legal field.

AILA 2019 [1] is one of the tracks in FIRE 2019, which aims to develop an automatic system that identifies a set of related prior cases as well as relevant statutes in the given situation. It consists of two tasks. Task 1 is to identify relevant prior cases for a given situation, and Task 2 is to identify relevant statutes for the situation.

The two tasks share 50 queries where each query describes a situation. In Task 1, there are 2,914 case documents as the candidates for searching, and in Task 2, there are 197 statutes from Indian law. Relevance labels for the first 10 queries are provided and can be used as training data, and the remaining 40 queries are used as test data. Each team can submit up to 3 runs for each task.

[1] Copyright © 2019 for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0). FIRE 2019, 12-15 December 2019, Kolkata, India

In our experiments, we model each task as a ranking problem. We utilize some classic retrieval models such as the LMIR, BM25 models but we also try to use auto-summarization and data processing techniques to further improve the ranking performance. Evaluated by P@10, MAP, BPREF, and RR, our three runs achieve the best performance in Task 2 and our best run was ranked at the 10th position in Task 1. We will introduce our methods and discuss the experiment results in the following sections.

2 Methods

Each task was treated as a ranking problem. We considered several classic retrieval models, including LMIR (Language Model for Information Retrieval) [7], VSM (Vector Space Model) [6], and BM25 [5]. We used varied settings and also tried to combine different retrieval models in different runs. In this section, we first introduce these retrieval models and then explain how we apply them to the two tasks.

Language Model for Information Retrieval (LMIR). Each document is considered as a language sample, and a query as a generation process [7]. The model ranks documents according to the probabilities of generating the query from the corresponding language models of documents, i.e. $P(q|d)$. Considering the unigram example, the probability can be written as

$$P(q|d) \propto \sum_{t \in q} P(t|d) \tag{1}$$

, where $P(t|d)$ is estimated based on term frequency. In the linear interpolation language model, $P(t|d)$ is estimated by

$$P(t|d) = \lambda P_{mle}\left(t|M_d\right) + (1-\lambda) P_{mle}\left(t|M_c\right) \tag{2}$$

, where $P_{mle}\left(t|M_d\right)$ denotes the term frequency of t in the document d, $P_{mle}\left(t|M_c\right)$ denotes the term frequency in the entire collection, and λ is a smoothing parameter ranging from 0 to 1. The bigram model can be combined with the unigram model as

$$P\left(w_{i-1}, w_i|d\right) = \mu \times P_1\left(w_i|d\right) + (1-\mu) \times P_2\left(w_{i-1}, w_i|d\right) \tag{3}$$

, where P_1, P_2 denote the unigram LM and bigram LM respectively, which can be estimated using formula (2), and μ also functions as a smoothing parameter. Considering $w_{i-1}w_i$ as the term t in formula (1), we can build a language model that considers both unigram and bigram.

Vector Space Model (VSM). The query and document are mapped to a latent space, and the similarity between the query and document are calculated by the vectors in the latent space. To be more specific, the query q and document d_j are represented as vectors, i.e. $\mathbf{q} = (\omega_{1,q}, \omega_{2,q}, ..., \omega_{N,q})$, $\mathbf{d_j} = \left(\omega_{1,d_j}, \omega_{2,d_j}, ..., \omega_{N,d_j}\right)$. Then the similarity can be calculated as a cosine value between the two vectors.

$$\cos(\mathbf{q}, \mathbf{d_j}) = \frac{\mathbf{q} \cdot \mathbf{d_j}}{\|\mathbf{q}\| \, \|\mathbf{d_j}\|} \tag{4}$$

In the classic VSM [6], the term-specific weights are represented as products of local and global parameters, i.e. $\omega_{t,d} = tf_{t,d} \cdot idf_t$, where $tf_{t,d}$ is the term frequency of term t in document d and idf_t is the inverse document frequency for term t. The inverse term frequency (idf) is a global parameter calculated based on the document collections: $idf_t = \log \frac{|D|}{|\{d' \in D | t \in d'\}|}$.

BM25 . Okapi BM25 [5] is a ranking function based on the probabilistic retrieval framework. It is widely used in kinds of search engines. The relevance score between the query and the document is computed by the following formula:

$$\sum_{i=1}^{n} idf\,(q_i) \cdot \frac{tf\,(q_i, d) \cdot (k_1 + 1)}{tf\,(q_i, d) + k_1 \cdot \left(1 - b + b \cdot \frac{|d|}{argdl}\right)} \cdot \frac{tf(q_i, q) \cdot (k_2 + 1)}{tf_{q_i,q} + k_2} \tag{5}$$

The k_1, k_2 and b are free parameters, usually chosen as $k_1 = k_2 = 1.0$, $b = 0.75$.

Mixed-size Bigram Model (MBM). MBM [2] is a variation of n-gram and tf-idf models. Firstly, the set of n-grams is obtained from query and document, denoted as $q_gramset$ and $d_gramset$ respectively. In MBM, we consider both unigram and bigram. Then, the relevance score between query and document is given by the following formula:

$$\frac{\sum_{\forall t} idf(t)}{I_q \times |q_{gramset}| + (1 - I_q) \times |d_{gramset}|}, t \in (q_{gramset} \cap d_{gramset}) \tag{6}$$

, where $0 \leq I_q \leq 1$ represents the relative significance of the query gram set.

2.1 Task 1: Identifying Relevant Prior Cases

In Task 1, we submit 3 runs which utilize LMIR, VSM, and a combination model (CM) of VSM and MBM, respectively.

Data Pre-processing. The query, which is a situation description, contains over 500 words on average, and the document contains over 3,000 words on average. Compared to the traditional ad-hoc search, the query in this task is much longer. We observe that the main legal issues of the first ten queries usually occur around the appeal process. Therefore, we select the sentences which contain "high court" as the key sentences, along with the former and latter sentences [2]. The query is made up of selected key sentences. We pre-process [3] all the queries and document in the following steps:

[2] If there are no sentences that contain "high court", we use the last four sentences
[3] We use the NLP tools produced by NLTK, http://www.nltk.org/ .

– Tokenize with RegexpTokenizer and ignore all of the punctuation and numeric.
– Make all of the words lowercase.
– Get POS tags by NLTK and ignore the conjunction, prepositions, determiners and list symbols.
– Ignore the English stop words.
– Do stemming using NLTK.

Run1: LMIR. We utilize the LMIR model with unigram and bigram in the first run. λ and μ are two parameters in this model, which both range from 0 to 1. We use a grid search with a step size of 0.1 for parameter tuning, based on Recall@20 and Recall@100 of the training data. As a result, we set $\lambda = 0.1$ and $\mu = 0.1$.

Run2: VSM. We get the *idf* based on all of the case documents. However, when we represent the document using a vector, the long text might weaken the representative power. Therefore, we only consider the last three paragraphs in each document. Then we use formula (4) to compute the similarity score between each query-document pair.

Run3: CM. In the last run, we utilize the linear combination of VSM and MBM, where $score_{CM} = \alpha \cdot score_{VSM} + (1 - \alpha) \cdot score_{MBM}$. The VSM score ($score_{VSM}$) is computed in run2, and we further compute $score_{MBM}$ with formula (6). There are two parameters, I_q in MBM and α, which both range from 0 to 1. Similarly, we use a grid search to select the best parameters and finally set $I_q = 0.8$ and $\alpha = 0.5$.

2.2 Task 2: Identifying Relevant Statutes

In this task, we submitted 3 runs which employed the LMIR, VSM and a combination model of VSM and BM25, respectively.

Data Pre-processing. The candidate documents in this task are statues, including titles and descriptions. There are two main differences from Task 1. One is that the document length is much shorter than that of Task 1, which is about 200 words on average. The other is that not only the issue but also the topic of the query situation contribute to the relevance. Therefore, instead of selecting the key sentences, we attempt to make use of the entire description. We consider two kinds of queries, one is to use the original situation text, denoted as q_{orl}, and the other is to use the summary (no more than 200 words) of situation description generated by TextRank [4] [4], denoted as q_{sum}. We concatenate the title and description of a statute as a candidate document. Then we apply the same data pre-processing steps listed in Section 2.1 to all the queries (q_{orl} and q_{sum}) and documents.

[4] https://github.com/PKULCWM/PKUSUMSUM

Run1: LMIR. The LMIR model in Task 2 is similar to that in Task 1, but we use the summary as the query. We search the parameters λ and μ according to Recall@10 and Recall@20, and finally set $\lambda = 0.1$ and $\mu = 0.1$.

Run2: VSM. We calculated the *idf* based on the statutes in this task. Since the document is quite short, we make use of the entire document text to get the document vector.

Run3: CM. We linearly combine the VSM with BM25 so that the relevance score is computed as $score_{CM} = \alpha \cdot score_{VSM} + (1 - \alpha) \cdot score_{BM25}$, where $score_{VSM}$ is given in run2 and $score_{BM}$ is calculated according to formula (5). We select the weighted parameter α with grid search based on Recall@10 and Recall@20, and set $\alpha = 0.7$, while for the parameters in BM25, we set $k_1 = k_2 = 1.0$ and $b = 0.75$.

3 Experiments and Results

Table 1 shows the results of our runs in Task 1 and Task 2, along with the results of some post experiments for Task 2. We made some mistakes when submitting the runs in Task 2. One was that we would like to set the $\alpha = 0.7$ in CM, but in the submitted run, the α was set as 0.3. The other was that we would like to use q_{sum} in all our three runs, but actually we used q_{orl} to calculate $score_{VSM}$ (the scores given by VSM in Run 2 and Run 3). Therefore, we conducted some post experiments with the labels and evaluation methods given by the organizers. Note that **CM*** here means to use the correct $\alpha = 0.7$.

Despite these mistakes, we achieved good results in Task 2, where our three runs were ranked as the top 3 runs among all the other runs. In the post experiments, we found that given the correct parameter settings, the combination of models can improve the performance in both tasks. Meanwhile, the performance of different models varies with tasks. For example, in Task 2, VSM performs quite well while in Task 1, it performs the worst. It suggests that different methodologies should be used for the precedent retrieval and statute retrieval.

In terms of methods to shorten the query length, we extract the key sentences through string matching. This decision is made based on our observation on only ten training queries, which might result in a weak generalization ability and therefore, hurt the models' performances in Task 1. In Task 2, we use auto summarization techniques, which seems to work well for LMIR and BM25 but have little effects on VSM.

4 Conclusion and Future Work

In this paper, we introduce the methods we employed in completing the AILA 2019 tasks. Our methods achieve the best ranking performance in Task 2, but do not work well in Task 1. This result suggests that we should use different methods in these two tasks and we need to further improve the models for the

Task ID	Method	P@10	MAP	BPREF	RR	Rank
Task 1	CM	**0.0425**	**0.0689**	**0.0434**	**0.121**	10
Task 1	LMIR	0.0375	0.0599	0.0316	0.149	12
Task 1	VSM	0.0225	0.0405	0.0221	0.095	17
Task 2	VSM (q_{orl})	0.0975	0.1566	0.0961	0.281	**1**
Task 2	CM ($q_{orl} + q_{sum}$)	0.09	0.1318	0.0742	0.247	2
Task 2	LMIR (q_{sum})	0.065	0.1115	0.0653	0.23	3
Task 2 (Post)	VSM (q_{sum})	0.0975	0.1404	0.0969	0.2576	–
Task 2 (Post)	CM* ($q_{orl} + q_{sum}$)	0.1025	**0.1742**	**0.1098**	0.3113	–
Task 2 (Post)	CM* (q_{sum})	**0.1125**	0.1550	0.1078	**0.3123**	–

Table 1. Evaluation Results in Task 1, Task 2 and post experiments for Task 2.

precedent retrieval task. Due to the limited size of data, we did not use neural models here and leave it as a research direction for further study.

5 Acknowledgement

This work is supported by the National Key Research and Development Program of China (2018YFC0831700) and Natural Science Foundation of China (Grant No. 61622208, 61732008, 61532011) .

References

1. Bhattacharya, P., Ghosh, K., Ghosh, S., Pal, A., Mehta, P., Bhattacharya, A., Majumder, P.: Overview of the FIRE 2019 AILA track: Artificial Intelligence for Legal Assistance. In: Proceedings of FIRE 2019 - Forum for Information Retrieval Evaluation (December 2019)
2. Carvalho, D.S., Nguyen, M.T., Tran, C.X., Nguyen, M.L.: Lexical-morphological modeling for legal text analysis. In: JSAI International Symposium on Artificial Intelligence. pp. 295–311. Springer (2015)
3. Mandal, A., Ghosh, K., Bhattacharya, A., Pal, A., Ghosh, S.: Overview of the fire 2017 irled track: Information retrieval from legal documents. In: FIRE (Working Notes). pp. 63–68 (2017)
4. Mihalcea, R., Tarau, P.: Textrank: Bringing order into text. In: Proceedings of the 2004 conference on empirical methods in natural language processing. pp. 404–411 (2004)
5. Robertson, S.E., Walker, S.: Some simple effective approximations to the 2-poisson model for probabilistic weighted retrieval. In: SIGIR94. pp. 232–241. Springer (1994)
6. Salton, G., Wong, A., Yang, C.S.: A vector space model for automatic indexing. Communications of the ACM **18**(11), 613–620 (1975)
7. Song, F., Croft, W.B.: A general language model for information retrieval. In: Proceedings of the eighth international conference on Information and knowledge management. pp. 316–321. ACM (1999)