FIRE2020 AILA Track: Legal domain search with minimal domain knowledge

Tobias Fink[a], Gabor Recski[a] and Allan Hanbury[a]

[a]TU Wien, Faculty of Informatics, Research Unit E-commerce

Abstract

We tackle task1 in the AILA 2020 shared task, where the goal is to retrieve precedent and statute documents related to a case document query in the Indian legal domain. We use BM25 with simple hyperparameter tuning and preprocessing for both precedent and statute retrieval and achieve a Mean Average Precision (MAP) of 0.1294 and 0.2619, respectively. We also experiment with removing frequent terms from the query as well as removing terms that produce high scores only in irrelevant documents, but both methods fail to improve the baseline results.

Keywords

information retrieval, legal domain, BM25

1. Introduction

In domain specific information retrieval (IR) each domain comes with its own challenges and its own language. Getting an understanding of the domain specific language and knowing which words and phrases can help to distinguish documents is important for IR, but unfortunately the intricacies of such a language are often difficult to understand and only known by domain experts. For example, in the case law system, there is the need to retrieve precedents and relevant statutes for legal documents, like cases. However, due to the length of such a document, it can contain passages about several topics, not all of which are helpful in distinguishing between documents, and might contain terms of which only some are related to relevant facts and rules.

In Task 1 of the FIRE2020 AILA track[1], the goal is to retrieve relevant precedent cases judged by the Indian Supreme Court and statutes from Indian law for queries consisting of legal case documents. A training set consisting of legal document queries as well as relevant and irrelevant precedent and statute documents is provided. It can be challenging for a non-expert to understand why a document is relevant or irrelevant for a particular query, because sometimes relevant and irrelevant documents seemingly deal with similar topics. This is more pronounced with longer documents, such as precedent case documents.

To tackle this task using only the relevance information and the text data of the provided training set, we use the well known BM25 document ranking algorithm[2] (implemented by the

Forum for Information Retrieval Evaluation 2020, December 16-20, 2020, Hyderabad, India

tobias.fink@tuwien.ac.at (T. Fink); gabor.recski@tuwien.ac.at (G. Recski); hanbury@ifs.tuwien.ac.at (A. Hanbury)

open-source Lucene-based search tool elasticsearch[1]) and determine hyperparameters for BM25 based on a random search on the training set. Further, we use simple heuristics to detect query terms that could be harmful to the desired search outcome and remove them from the query. The heuristics decide whether a term should be removed based on the frequency of each term in the corpus and the BM25 scores of each query term across a set of relevant and irrelevant documents.

2. Dataset and Related Work

The dataset is partially taken from last year's AILA2019 [3] and consists of 3.257 prior case documents (2914 old, 343 new) and 197 statute documents as well as 50 training queries for which the relevance of precedents and statutes is known and 10 test queries. The queries consist of a paragraph of raw text and are as such rather different from search queries typically entered into web search engines, which are usually much shorter. The mean (standard deviation) of relevant documents per query is 3.9 (3.82) and 4.42 (0.67) for precedents and statutes, respectively.
For the AILA2019, there where many submissions successfully employing BM25 in some form or another. One of the top performers in both precedent and statute retrieval, Zhao et al. [4] employ a new relevance score created by calculating BM25 on a filtered query document and an unfiltered query document and adding the two scores. The filtering is done by ranking the query terms according to their IDF-scores and taking the top 50 highest scoring terms. Additionally, they also experiment with a Word2Vec based similarity function, which works well for statute retrieval but not precedent retrieval. Similarly, Gao et al. [5] submitted runs using TF-IDF or Textrank to first extract the top 60 to 80 words from the query and using a Vector Space Model (VSM), BM25 and a Language Model (LM) for retrieval. For Task 1 the TF-IDF based query extraction paired with the VSM achieves 2nd place, followed by TF-IDF paired with BM25 achieving 4th place. They did not submit any runs for Task 2. For Task 1, Shao et al. [6] extract sentences containing the phrase "high court" as key sentences and utilize VSM, LM and a VSM + Mixed-Size Bigram Model combination but only achieve rank 10 for this task. For Task 2, they use the entire description and utilize VSM, LM and a VSM + BM25 combination. They achieve rank 1 (VSM), rank 2 (VSM+BM25) and rank 3 (LM) for statute retrieval. While in these works the hyperparameters of BM25 k_1 and b are set to static values, the choice which values should be selected is also topic of research in IR. For example, in Lipani et al. [7], b is instead calculated based on the mean average term frequency of a collection. Due to the overall good performance of BM25 based methods, we also opt to experiment with this retrieval method.

3. Methodology

To retrieve data from the corpus, we create two indices using elasticsearch[2], one for precedents and one for statutes. While elasticsearch has their own stack of text pre-processing and analysis tools, we opt to perform our text preprocessing outside of elasticsearch, since some of the desired functionality, like lemmatization, is not supported by elasticsearch. Instead we use the

[1] https://github.com/elastic/elasticsearch
[2] https://www.elastic.co/elastic-stack

open source natural language processing library spacy[3] to tokenize the document text. We further clean the text by removing punctuation tokens, numbers and typical English stopwords. Finally, the tokens for each document are lemmatized, lowercased and then added to a single indexed field. We generate our queries by applying the same procedure to the query documents, but since the query documents can be very long and occasionally exceed the elasticsearch max clause limit of 1024, we remove all duplicate tokens from the resulting list of tokens.

3.1. Ranking Method

We score the documents using the commonly used Okapi BM25 ranking function (as is implemented in elasticsearch), which is calculated using the following formula:

$$BM25(D,Q) = \sum_{q \in Q} IDF(q) \cdot \frac{tf(q,D) \cdot (k_1 + 1)}{tf(q,D) + k_1 \cdot (1 - b + b \cdot \frac{|D|}{avgdl})} \tag{1}$$

where D is the document to be scored, Q is the query, q is a query term/token occurring in the query, $tf(q,D)$ is the term frequency of token q in document D, $|D|$ is the length of the document in tokens and $avgdl$ is the average document length for documents in the collections. Further, k_1 and b are hyperparameters and $IDF(q)$ is the inverse document frequency calculated by this formula:

$$IDF(q) = ln(\frac{N - n(q) + 0.5}{n(q) + 0.5} + 1) \tag{2}$$

where N is the total number of documents and $n(q)$ is the number of documents containing query term q.

3.2. Hyperparameter Search

Since we do not know what the best hyperparameter values for k_1 and b are for our two tasks, we decided to experiment with the selection of the values. Instead of taking the often used values of $k_1 = 1.2$ and $b = 0.75$, we do a random search to determine the values that best fit the collection. We repeatedly select random values from an interval of $[1.2, 2.0]$ for k_1 and $[0.0, 1.0]$ for b, run our 50 training queries and evaluate the results. We take the values that resulted in the best performance after 30 repetitions as our final values. We use the mean average precision (MAP) metric to quantify the performance of an iteration, shown in the following formula:

$$MAP(Q) = \frac{\sum_{q \in Q} AvgP_q}{|Q|} \tag{3}$$

where Q is the set of training queries, q is a single query and $|Q|$ is the number of queries. Further, the average precision of a query $AvgP_q$ is calculated with the following formula:

$$AvgP_q = \frac{\sum_{k=1}^{n} P_q(k) \cdot rel_q(k)}{|R_q|} \tag{4}$$

[3] https://spacy.io/

where n is the number of retrieved documents, $P_q(k)$ is the Precision @ k for query q, $|R_q|$ is the number of relevant documents for query q and $rel_q(k)$ is 1 if document at rank k is relevant otherwise 0.

3.3. Finding problematic terms

As we are unfamiliar with the Indian legal domain and we consequently do not know the typical keywords and phrases of the domain, we attempt to gain some insight into the domain using the relevance judgements that we have. We looked at the BM25 scores assigned to individual query terms q (see Formula 1) of relevant and irrelevant documents and noticed a few issues:

- If there are enough query terms with a high term frequency and a high document frequency like "court", this can cause an irrelevant document to be ranked higher than a relevant one.
- There are query terms that have a high score in irrelevant documents, but not in relevant ones, because they either are less frequent in relevant documents or do not occur there.
- It appears that some documents that are relevant for poor-performing queries are suppressed by irrelevant documents. In these documents most high-scoring terms also appears in irrelevant documents and have a higher term frequency (relative to document length) there, while at the same time containing no high-scoring terms that are unique to them.

Based on these findings, we develop a heuristic to detect query terms that would cause irrelevant documents to be ranked higher than relevant ones. These "additional stopwords" detected by the heuristic are then removed from the query and every remaining term of the query is treated as a search term. We experiment with the following approaches for detecting these "additional stopwords" in the query:

Word Count: We filter out the most frequent words in the corpus. We preprocess precedents and statutes and count how often each term occurs in each respective corpus. Using this information, we add the 200 most frequent terms to our list of "additional stopwords". This is done for precedent and statute documents separately.

False Friends: We measure the BM25 scores assigned to individual query terms q (see Formula 1) of our training queries and compare the results for relevant and irrelevant documents. Using the static hyperparameters $k1 = 1.2$ and $b = 0.75$, we calculate a ranking for each training query. Then we retrieve[4] the scores of each term for each relevant document and for the first 100 irrelevant documents. For each query term q across all training queries, we calculate a classification q_r and q_{ir} from the maximum score of that token for all retrieved relevant and irrelevant documents respectively. The classification q_r and q_{ir} for a token is either 'Not Found' if the query token was not found in the retrieved documents, 'Low' if the maximum score was at or below the threshold t and 'High' if the maximum score was above t. We add those tokens q

[4]Using the elasticsearch explain functionality

Run_ID	MAP	BPREF	recip_rank	P @ 10
Precedent Retrieval				
basic	**0.1294**	**0.0737**	**0.1915**	**0.07**
false_friends	0.1133	0.0687	0.1873	0.05
word_count	0.1271	0.0728	0.1891	0.06
Statute Retrieval				
basic	**0.2619**	0.2033	**0.4855**	0.13
false_friends	0.2316	0.1855	0.3814	0.1
word_count	0.2574	**0.214**	0.3946	**0.14**

Table 1
Results of our runs on the test set, showing the best among our results in bold.

that have a label q_{lir} = 'High' **AND** q_r = 'Low' or 'Not Found' to our stopword list. Based on a separate grid search experiment, we set the parameter $t = 1.5$.

However, tests with these methods on the training data using cross-validation showed that they did not consistently improve the retrieval results. Due to a lack of further development time, we submitted these methods as runs to measure their performance on the test set.

4. Results

We submitted three runs for precedents and statutes each. The **basic** run only performs the hyperparameter search, while the **word_count** and **false_friends** run both include the hyperparameter search and their respective method of detecting additional stopwords. The results of these runs are shown in Table1 and show that among our runs the basic method generally achieves the best results. Compared to the other groups, our best method can be found around the middle of the ranking. The best overall precedent retrieval MAP was **0.1573** (run UB-3) and best overall statute retrieval MAP was **0.3851** (run scnu_1).
This tells us that using BM25 with some preprocessing and hyperparameter tuning is still a good start when trying to work with a new domain. However, our method of removing additional stopwords from the query proved detrimental and other methods of extracting keywords from document queries should be considered. Removing the most frequent terms from a query either does not retrieve more relevant documents or makes more relevant documents more difficult to retrieve, hinting that these tokens still carry some useful information even if they are very frequent. Also, the way we remove terms based on their BM25 scores might be very prone to overfitting on the training set. It might still be possible that these removed terms are important for unknown relevant documents of unknown queries. A better way to work with such a 'High/Low' classification might be to assign higher weights to (boost) query terms of which we know they score highly in relevant documents.

5. Conclusion

We conclude that BM25 can be a good starting point when working with an unfamiliar domain. In the Indian legal domain and with little hyperparameter tuning, it achieves a MAP about 18% lower than the top result on precedent retrieval and about 32% lower than the top result on statute retrieval. We attempted utilizing the word count and the BM25 query token scores of training queries to detect unimportant or harmful tokens as additional stopwords. However, removing either from the query document did not improve results.

Acknowledgments

Project partly supported by BRISE-Vienna (UIA04-081), a European Union Urban Innovative Actions project.

References

[1] P. Bhattacharya, P. Mehta, K. Ghosh, S. Ghosh, A. Pal, A. Bhattacharya, P. Majumder, Overview of the FIRE 2020 AILA track: Artificial Intelligence for Legal Assistance, in: Proceedings of FIRE 2020 - Forum for Information Retrieval Evaluation, Hyderabad, India, 2020.

[2] S. E. Robertson, S. Walker, S. Jones, M. M. Hancock-Beaulieu, M. Gatford, Okapi at TREC-3, Nist Special Publication Sp 109 (1995) 109. Publisher: NATIONAL INSTIUTE OF STANDARDS & TECHNOLOGY.

[3] P. Bhattacharya, K. Ghosh, S. Ghosh, A. Pal, P. Mehta, A. Bhattacharya, P. Majumder, FIRE 2019 AILA Track: Artificial Intelligence for Legal Assistance, in: Proceedings of the 11th Forum for Information Retrieval Evaluation, 2019, pp. 4–6.

[4] Z. Zhao, H. Ning, L. Liu, C. Huang, L. Kong, Y. Han, Z. Han, FIRE2019@ AILA: Legal Information Retrieval Using Improved BM25., in: Working Notes of FIRE 2019 - Annual Meeting of the Forum for Information Retrieval Evaluation, CEUR Workshop Proceedings, volume 2517, Kolkata, India, 2019, pp. 40–45.

[5] J. Gao, H. Ning, H. Sun, R. Liu, Z. Han, L. Kong, H. Qi, FIRE2019@ AILA: Legal Retrieval Based on Information Retrieval Model., in: Working Notes of FIRE 2019 - Annual Meeting of the Forum for Information Retrieval Evaluation, CEUR Workshop Proceedings, volume 2517, Kolkata, India, 2019, pp. 64–69.

[6] Y. Shao, Z. Ye, THUIR@ AILA 2019: Information Retrieval Approaches for Identifying Relevant Precedents and Statutes., in: Working Notes of FIRE 2019 - Annual Meeting of the Forum for Information Retrieval Evaluation, CEUR Workshop Proceedings, volume 2517, Kolkata, India, 2019, pp. 46–51.

[7] A. Lipani, M. Lupu, A. Hanbury, A. Aizawa, Verboseness fission for BM25 document length normalization, in: Proceedings of the 2015 International Conference on the Theory of Information Retrieval, 2015, pp. 385–388.