University of Amsterdam at the TREC 2007 Legal Track

Avi Arampatzis[1] **Jaap Kamps**[1,2] **Marijn Koolen**[1] **Nir Nussbaum**[2]

[1] Archives and Information Studies, Faculty of Humanities, University of Amsterdam
[2] ISLA, Informatics Institute, University of Amsterdam

Abstract: In this paper, we document our official submissions to the TREC 2007 Legal track. Our main aims were two-fold: First, we experimented with using different query formulations trying to exploit the verbose topic statements. Second, we analysed how ranked retrieval methods can be fruitfully combined with traditional Boolean queries. Our main findings can be summarized as follows. First, we got mixed results trying to combine the original search request with terms extracted from the verbose topic statement. Second, by combining Boolean and ranked retrieval allows us to get the high recall of the Boolean retrieval, whilst precision scores show an improvement over both the Boolean *and* the ranked retrieval runs.

1 Introduction

As part of the TREC 2007 Legal track, we experimented with different query formulations and run combinations. Since the focus in the Legal track ad hoc evaluation is on recall oriented measures, we investigated methods to increase recall by combining result lists based on different query representations. We also analysed differences between our ranked retrieval runs and the Boolean reference run, and investigated ways to fruitfully combine the strengths of both approaches.

The rest of this paper is organized as follows. In Section 2, we detail the experimental set-up. In Section 4, we discuss our official submissions, results, and additional experiments. Finally, we summarize our findings in Section 5.

2 Experimental Set-up

2.1 Retrieval set-up

Our retrieval system is based on the Lucene engine version 1.9 [3].

Indexes The Legal track uses the a test collection, containing 6,910,192 documents (∼58Gb uncompressed). The documents are legal documents, on issues of the tobacco industry. The documents are all in XML format, containing limited meta-data.

For Trec Legal 2007, we created two separate indexes, as following:

Full-text the full textual content of the documents, including the meta-data tags, as is (∼37GB);

Text-only the text inside the tags, not including the tags (∼33GB).

During indexing, we captured the document ID and stored it in the index. In tokenization, we removed the common stopwords and stemmed using the Snowball stemming algorithm [4].

The original corpus contains almost 7 million documents in 650 files and is over 58Gb uncompressed data. The filename convention is `iitcdip.x.y.xml`, where `x` is a letter a-z (excluding s) and `y` is a letter a-z. As a first step we decided to create 25 partial indexes for each of the two indexing choices mentioned above, for example: index `a` indexed the files `iitcdip.a.a.xml` to `iitcdip.a.z.xml`. We chose to index in chunks because a single indexing process would have taken relatively very long time. Indexing in chunks enabled us to use six computers concurrently, each indexing different chunks. Since accessing multiple indexes is substantially slower, we eventually merged the 25 indexes into one index, an action that Lucene is handling straightforwardly.

Retrieval model For ranking, we use a vector-space retrieval model. Our vector space model is the default similarity measure in Lucene [3], i.e., for a collection D, document d and query q:

$$sim(q,d) = \sum_{t \in q} \frac{tf_{t,q} \cdot idf_t}{norm_q} \cdot \frac{tf_{t,d} \cdot idf_t}{norm_d} \cdot coord_{q,d} \cdot weight_t,$$

where

$$tf_{t,X} = \sqrt{\text{freq}(t,X)}$$

$$idf_t = 1 + \log \frac{|D|}{\text{freq}(t,D)}$$

$$norm_q = \sqrt{\sum_{t \in q} tf_{t,q} \cdot idf_t{}^2}$$

$$norm_d = \sqrt{|d|}$$

$$coord_{q,d} = \frac{|q \cap d|}{|q|}$$

3 Experiments

3.1 Runs

This was our first participation in Trec Legal, and we only managed to complete a single run by the deadline, coded `catchup0701p`. All the other runs are post-submission experiments and have not been included in the pooling process.

We made runs using the search request as stated in the *RequestedText* tag:

Full-text Runs on the *Full-text* index. In this run, we used the Snowball stemming algorithm on the *RequestedText* meta-data in the query.

This is our official submission `catchup0701p`, however, we discovered a processing error so we show here the results for the corrected run.

Text-only The same as *Full-text*, but it runs on the *Text-only* index.

The Legal Track topics have very lengthy topic descriptions providing a range of background information on the topic of request. Hence, we tried to extract potentially useful terms from it. Specifically, we decided to select only those terms that are most characteristic for a single topic, with reference to the whole topic set. That is, the terms that best distinguish the topic at hand from the other topics in the topic set. For this we used a variant of the parsimonious language modeling techniques [2], and create a query by selecting the 25 terms that are most characteristic for the topic. For example, topic 52 reads:

```
Please produce any and all documents
that discuss the use or introduction of
high-phosphate fertilizers (HPF) for the
specific purpose of boosting crop yield
in commercial agriculture.
```

and the 25 selected terms are:

```
hpf sugar mh vsf gcc valhalla candy
phosphate plaintiffs its fertilizers
crop high gladsheim beet yield community
groundwater health death use fertiliz
contamination phosphat cause
```

We use the selected terms in the following ways:

Table 1: Statistics over judged and relevant documents per topic.

	nr. of	per topic				
	topics	min	max	median	mean	st.dev
judged	43	488	1,000	499	567.53	164.84
relevant	43	10	391	72	101.02	97.76
B	43	103	22,518	2,665	5,004.02	6,156.75

SelectedTerms The query string fed into Lucene is the original query (the *RequestedText* tag) appended by the most significant 25 terms in the background information supplied in the file $fullL07\_v1.xml$. Duplicate terms are removed.

CombiTextSelectedTerms A combination of **Text-only** and **SelectedTerms**. We use the standard combination method CombSUM [1] and combine full length runs, without normalizing the scores.

4 Results

4.1 Topics and judgments

The results are based on the qrels over 43 topics. Statistics of the assessments are shown in Table 1. We include the number of results of the negotiated Boolean query ("B" for short), since it plays an important role in the recall oriented measures used. It is striking that the B number is orders of magnitude larger than the number of known relevant documents. Moreover, there is no significant correlation between B and the number of relevant documents (Pearson r = 0.059). There is a significant correlation (0.55) between the number of judged and number of found relevant documents, which is not unexpected.

4.2 Runs

Table 2 shows the results for the Legal track (all scores based on l07_eval v1.0). First, we look at (the corrected version of) our official submission, **Full-text**, and compare it to the similar **Text-only** run. The **Full-text** run scores marginally better on bpref, but the **Text-Only** run scores marginally better on all other measures including the main measure estimated recall at B.

Second, we look at the **SelectedTerms** run. We see a drop in performance for all measures. This is not unexpected: the selected terms from the complete topic statement are less focused on the topic. Nevertheless, the run may have picked up documents that are missed by the original query or improve the ranking of retrieved relevant documents. Can we use these results to improve recall in our original run? We combine the results from the two runs **Text-only** and **SelectedTerms** using standard CombSUM. That is, for a result r_i, we compute the combined score

Table 2: Results for the Legal track (using `l07_eval` v1.0).

Run	MAP	bpref	P10	num_rel_ret	recallB	est_RB
Full-text	0.0878	0.3266	0.2837	3338	0.4792	0.1448
Text-only	0.0880	0.3255	0.2860	3339	0.4835	0.1548
SelectedTerms	0.0355	0.2619	0.1070	2522	0.3173	0.0772
CombiTextSelectedTerms	0.0846	0.3302	0.2698	3306	0.4841	0.1447
`refL07B`	0.0167	0.2902	0.0209	2145	0.4864	0.2158
CombiTextRef	0.1181	0.3842	0.3209	3553	0.4864	0.2158

$S_{CombSum}(r_i) = S_A(r_i) + S_B(r_i)$. The results are mixed. For the **CombiTextSelectedTerms** run, we see a drop in MAP and Precision10 when compared to the best individual run, but an increase in bpref. We see a minimal gain in recallB, but a loss of estimated recall at B.

4.3 Combining ranked and Boolean retrieval

We conducted further experiments trying to shed light on the relative strength and weaknesses of our ranked retrieval methods versus the Boolean reference run.

First, we looked at the reference Boolean run `refL07B` which has unranked set results (all selected documents have a RSV of 1). As Table 2 shows, this results indeed in very poor MAP and p10 scores. The bpref score is comparable to the scores of ranked retrieval (this is also clearly signalling that bpref should not be treated as an approximation of traditional MAP). In terms of recall, the reference run is retrieving fewer relevant documents overall (but has only B results per topic, whereas our runs have up to 25,000), has slightly better recall at B, and has much better estimated recall at B. Summarizing, the reference run has unimpressive precision but very good recall.

Is there a way to combine the strength of ranked and Boolean approaches? What we did is the following. Recall that `refL07B` run assigns a score of 1 to every document. Our runs score in the range $[0, 1]$. Now, consider what happens if we combine the reference run with one of our runs: it will first have all documents of the reference run, but ranked by our retrieval score, and then have the remaining documents from our run, again ranked by our retrieval score. The run **CombiTextRef** in Table 2 combines the **Text-only** run with the `refL07B` run. What we see is that, indeed, the recall at B and estimated recall at B of the Boolean run are preserved. However, the MAP, bpref, and p10 scores even improve over the scores from the ranked retrieval run alone. Summarizing, combining Boolean and ranked retrieval allows us to get the best of both worlds: the high recall scores of the Boolean run are maintained, while the MAP and precision scores show an improvement over both the Boolean *and* the ranked retrieval run.

5 Conclusions

In this paper, we documented our official submissions to the TREC 2007 Legal track. Our participation in the Legal Track was driven by two main aims: First, we experimented with using different query formulations trying to exploit the verbose topic statements. Second, we analysed how ranked retrieval methods can be fruitfully combined with traditional Boolean queries.

Our initial findings can be summarized as follows. First, we got mixed results trying to combine the original search request with terms extracted from the verbose topic statement. Second, by combining Boolean and ranked retrieval allows us to get the high recall of the Boolean retrieval, whilst precision scores show an improvement over both the Boolean *and* the ranked retrieval runs.

Acknowledgments This research was supported by the Netherlands Organization for Scientific Research (NWO, grant # 612.066.513, 639.072.601, and 640.001.501), and by the E.U.'s 6th FP for RTD (project MultiMATCH contract IST-033104).

References

[1] E. Fox and J. Shaw. Combination of multiple searches. In D. Harman, editor, *The Second Text REtrieval Conference (TREC-2)*, pages 243–252. National Institute for Standards and Technology. NIST Special Publication 500-215, 1994.

[2] D. Hiemstra, S. Robertson, and H. Zaragoza. Parsimonious language models for information retrieval. In *Proceedings of the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 178–185. ACM Press, New York NY, 2004.

[3] Lucene. The Lucene search engine, 2007. http://lucene.apache.org/.

[4] Snowball. Stemming algorithms for use in information retrieval, 2007. http://www.snowball.tartarus.org/.