Initial Results with Structured Queries and Language Models on Half a Terabyte of Text

Kevyn Collins-Thompson Paul Ogilvie Jamie Callan
Language Technologies Institute
School of Computer Science
Carnegie Mellon University
5000 Forbes Avenue, Pittsburgh, PA 15213
{kct, pto, callan}@cs.cmu.edu

1. INTRODUCTION

The CMU Distributed IR group's experiments for the TREC 2004 Terabyte track are some of the first to use Indri, a new indexing and retrieval component developed by the University of Massachusetts for the Lemur Toolkit [2]. Indri combines an inference network with a language-modeling approach and is designed to scale to terabyte-sized collections.

Our goals for this year's Terabyte track were modest: to complete a set of simple baseline runs successfully using the new Indri software, and to gain more experience with Indri's retrieval model, the track's GOV2 corpus, and terabyte-scale collections in general.

2. COLLECTION AND TASK SUMMARY

The Terabyte track this year used the GOV2 corpus, which is made up of about 25 million Web documents crawled from the `.gov` Web domain, comprising about 426 Gb of document source.

The task for the Terabyte track is ad-hoc retrieval. There are 50 queries or 'topics'. Each topic is presented in the usual TREC format, as a tagged document with three fields that summarize the information need at different levels of detail: a short 'title' consisting of just a few terms, a longer, more detailed 'description' field, and a multi-sentence 'narrative' field.

3. INDEXING ENVIRONMENT

We indexed all of the GOV2 corpus documents, submitting complete documents to the indexer. We did no special processing for titles or other document structure. After running each document through Indri's HTML parser and stripping out tags, terms were normalized by removing punctuation, converting to lowercase, and performing Krovetz stemming [6]. Indri has a utility to extract anchor text, but a working version of this was not available at the time of our experiments, so no anchor text (or other link-derived features) was included. We did not use an index-time stop list or acronym list.

We partitioned the GOV2 corpus into six roughly equal-sized pieces and built a separate index for each subset. There were two reasons for doing this. First, it was our intent to distribute retrieval for the corpus across multiple machines: Indri has a facility for transparently federating retrieval results across the network. Second, it helped reduce the time to find and fix problems with interim Indri releases (typically, pathological HTML cases that would cause the parser to crash).

For various reasons, we ended up using a single PC for both indexing and retrieval (though not at the same time). The PC had dual Xeon 3.2GHz CPUs with 2Gb RAM running Red Hat Linux release 9. The file system used an ACNC SATA Raid array, specifically Jetsor III Raid/150869 with 128mb dual SCSI host 16 slots, with 8x250GB RAID-5 drives in the box and 6x146GB RAID-5 disks internal to the PC. All disks were 7200 RPM.

We used a modified version of Indri build 20040830-1620 to build the index set. (The changes were to work around various parsing bugs which have since been fixed.) Total indexing time was 20.1 hours and the resulting index files were 138 Gb in size. The index referenced 25,205,168 documents. Indri also generated a compressed version of the tokenized collection, which for the GOV2 corpus was 107 Gb in size.

4. RETRIEVAL ENVIRONMENT

4.1 Topic Processing and Query Formation

We converted topics to Indri structured queries as follows.

For title-only runs, we parsed each title field into its constituent unigrams and removed stop words using a stop list of 447 words derived from the standard list used with INQUERY [5]. The remaining terms were simply combined using the Indri #weight operator, with all terms given equal weight. (These prior weights are in addition to any term weighting done by the retrieval model.)

For runs using all three topic fields, we tried a very simple heuristic term matching scheme to identify unigrams,

bigrams, and trigrams that recurred across fields – the idea being that concepts occurring in multiple fields would be more likely to be central to the information need. There were three phases: 1) extracting candidate terms (unigrams, bigrams, and trigrams) from each field, 2) matching up candidates across fields while computing a weighted score for each candidate, and 3) selecting the top *N* candidates with scores above an empirically derived threshold. For scoring, fields were associated with weights giving a rough prior estimate of their importance, as follows: title: 0.50, narrative: 0.35, and description: 0.15. The combined score for a candidate term was obtained by summing the weights for the fields in which it occurred. Scores for all candidates were then normalized so that the highest-scoring candidate(s) were given a weight of 1.0. We used the same term selection threshold of 0.200 for all our experiments, since this value seemed to give the best relative performance in most cases. While our algorithm here was simplistic, it may be easily generalized to a more powerful translation model in which approximate term matching is done with scores obtained via a translation cost function.

The final all-field structured queries were formed by combining the selected candidate terms and their scores with the #weight operator. Within the #weight operator, phrases were mapped to proximity operators. We tried different window sizes ranging from 3 words up to 20 words. We settled on an unordered proximity window of 8 words for both bigrams and trigrams since this appeared to give slightly better accuracy in the cases we tried. An example of the resulting query is shown in Figure 1.

```
<top>
<num> Number: 452
<title> do beavers live in salt water
<desc> Description:
Describe the normal habitat for
beavers; note exceptions, if any.
<narr> Narrative:
Relevant documents describe the
habitat range as well as references to
specific areas and bodies of water.
</top>

#weight(
   1.000 beavers
   0.695 habitat
   0.571 water
   0.305 #uw8(beavers live)
   0.305 #uw8(live salt water)
   0.305 #uw8(beavers live salt)
   0.305 #uw8(live salt)
   0.305 #uw8(salt water)
   0.305 salt
   0.305 live
   )
```

Figure 1: Original TREC topic 452 (top), and corresponding structured query (bottom)

4.2 Language Model Smoothing

We used Dirichlet smoothing [8] for all submitted runs, with $\mu = 2500$, using the WT10g collection [3] to estimate optimal values for μ and other retrieval parameters. Indri allows different smoothing settings for term and window language models, but for our baseline we used the same setting for all language models. We also looked briefly at learning query-specific μ values (see Section 5).

More details on the Indri retrieval model are available in the UMass TREC 2004 paper [1].

4.3 Query Expansion

We used the default pseudo-relevance feedback algorithm in the Indri *runquery* utility (as of build 20040909-1720) to perform query expansion. This algorithm is a variant of a language-modeling approach to pseudo-feedback described by Lafferty and Zhai [7]. The final query is a weighted combination of the original and expanded queries, which in our experiments were weighted equally. We filtered the expansion terms by using a modified stop list that included Web-specific noise terms such as 'pdf', 'http', 'www', and so on. The best performance on WT10g given the other query processing parameters was obtained using the document count ('docs') and term count ('terms') parameters shown in Table 1.

5. EXPERIMENTS

We submitted two title-only runs, with and without pseudo-relevance feedback (cmutufs2500 and cmutuns2500 respectively), and one run using all topic fields with feedback (cmuapfs2500). A summary of the runs and their performance is given in Table 1. Our best performing run was cmuapfs2500, which ranked 3rd overall out of 71 runs in terms of MAP, R-precision, and bpref [4] scores.

Results for the WT10g collection on topics 451-550 are also given, in Table 2, showing the effects of adding various enhancements to the baseline method. The most significant improvements came from switching to Dirichlet smoothing with tuned μ parameter, and using terms from all fields instead of just the title. The best performing run used all fields, unigram terms, Dirichlet smoothing, and pseudo-feedback and obtained a MAP of 0.2524. Our heuristic phrase selection algorithm appeared to give slightly worse results than using unigrams. Investigating the reasons for this and finding improvements to our topic analysis, is the subject of future work.

We did some preliminary investigation into varying μ for each query as a function of simple features such as number of non-stopwords and mean log frequency of the query terms in general English, but the results were inconclusive.

Run	Fields Used	Terms	QE parameters	MAP
cmutuns2500	Title	Unigrams	No expansion	0.2071
cmutufs2500	Title	Unigrams	docs=5, terms=10	0.2481
cmuapfs2500	All	Phrases	docs=5, terms=20	0.2843

Table 1: Submitted runs for the GOV2 corpus using TREC topics 701-750 and top 10,000 documents. All runs used Dirichlet smoothing with $\mu = 2500$.

We also calculated an oracle run for the all-field topics by selecting the individual values for μ giving the best performance for each query. (μ was varied from 500 to 5000 in increments of 500.) This oracle run obtained a MAP of 0.2618. The comparable run with best fixed μ=2500 obtained a MAP of 0.2304.

6. CONCLUSIONS

The GOV2 collection was the largest processed by the CMU DIR group to date. We achieved our main goal for this track, which was to obtain a set of baseline results with Indri, a new indexing and retrieval component in the Lemur Toolkit [2]. These results made basic use of Indri's combined language modeling-based retrieval and inference network features. Indri provides a much richer set of query operators than we used for our experiments. For example, it has the ability to build language models for arbitrary fields, including fields for document structure. This should prove quite useful for many IR tasks and future work.

The only significant problem we encountered was the slow speed of long, expanded queries on a collection of this size, with such queries taking several minutes to complete. We believe this problem had two causes: first, because of system constraints, we ended up using an inefficient configuration, i.e. federated retrieval with multiple collection partitions on a single machine instead of multiple machines. Second, some significant speed improvements have since been made in Indri's query processing. We expect speed to be less of an issue in future experiments. Other problems, such as handling the wide range of HTML in such a large corpus, were relatively minor and an expected part of adopting early builds of complex software.

Overall, Indri proved to be reliable and scalable for this task, and we believe it represents a promising new tool for future large-scale retrieval experiments.

	Title	All fields
Baseline: Unigrams, no QE, Jelinek-Mercer smoothing, ($\lambda = 0.4$)	0.1290	0.1734
+ Dirichlet smoothing Best fixed $\mu = 2500$	0.2016	0.2376
+ Phrases (bi-grams, tri-grams)	0.1988	0.2188
+ QE (docs = 5, terms = 10) w/ Unigrams	0.2162	0.2506
QE (docs = 5, terms = 20) w/ Phrases	0.2102	0.2304
QE (docs = 5, terms = 20) w/ Unigrams	0.2160	0.2524

Table 2: Mean average precision results on WT10g, using TREC topics 451 – 550 and top 1000 documents.

7. ACKNOWLEDGEMENTS

This work was supported by Dept. of Education grant R305G03123. Any opinions, findings, conclusions, or recommendations expressed in this material are the authors', and do not necessarily reflect those of the sponsors. The authors thank Trevor Strohman and Don Metzler for their extensive help with Indri.

8. REFERENCES

[1] N. Abdul-Jaleel, J. Allan, W.B. Croft, F. Diaz, L. Larkey, X. Li, D. Metzler, M.D. Smucker, T. Strohman, H. Turtle, C. Wade. UMass at TREC2004: Notebook. TREC 2004 Conference Notebook, NIST Special Publication, 2004.

[2] J. Allan, J. Callan, K. Collins-Thompson, B. Croft, F. Feng, D. Fisher, J. Lafferty, L. Larkey, T. N. Truong, P. Ogilvie, L. Si, T. Strohman, H. Turtle, and C. Zhai. The Lemur toolkit for language modeling and information retrieval. `http://www.cs.cmu.edu/~lemur`

[3] P. Bailey, N. Craswell, and D. Hawking. Engineering a multi-purpose test collection for Web retrieval experiments. *Information Processing and Management*, 39(6): 853-871, November 2003.

[4] C. Buckley and E. M. Voorhees. Retrieval evaluation with incomplete information. *Proc. of SIGIR 2004*, Sheffield, U.K. pp. 25 – 32.

[5] J. P. Callan, W.B. Croft, and S. M. Harding. The INQUERY retrieval system. *Proceedings of the Third International Conference on Database and Expert Systems Applications*, Valencia, Spain, 1992, pp. 78 – 83.

[6] R. Krovetz. *Word Sense Disambiguation for Large Text Databases*. Doctoral thesis, Univ. of Mass., Amherst, 1995.

[7] J. Lafferty and C. Zhai. Document language models, query models, and risk minimization for information retrieval. *Proc. of SIGIR 2002*, New Orleans, U.S.A., pp. 111 – 119.

[8] C. Zhai and J. Lafferty. A study of smoothing methods for language models applied to ad-hoc information retrieval. *Proc. of SIGIR 2002*, New Orleans, U.S.A. pp. 334 – 342.