CSUSM at TREC2005: Genomics and Enterprise Track

Rocio Guillén
Computer Science Department, California State University San Marcos
email:rguillen@csusm.edu

Abstract

In this paper we report on the approach, experiments and results for the Enterprise Track and the Genomics Track. We participated in the adhoc and one of the categorization tasks for the Genomics track. For the enterprise track we participated in the known-item search. We ran experiments using Indri [1], which combines inference networks with language modeling, for the adhoc and the known-item search tasks. For the categorization task we filtered the documents in different stages using decision trees.

1. Retrieval Engine

Indri is a new language modeling retrieval engine developed at UMass [1] derived from the Lemur project. It was written to handle question answering and web retrieval tasks using large corpora. It has been used in the Terabyte, Novelty, and HARD tasks [2]. The engine is implemented using C++ and runs in different platforms. We installed Indri (Version 2.0) on Linux.

Indri can be used to build an index/repository with the application *buildindex*, run queries using the index built with the application *runquery*, and the *indrid* application is a repository server. Buildindex can build repositories from TREC formatted documents (e.g., terabyte track), HTML documents, text documents and PDF files. It also handles HTML/XML documents. Runquery evaluates queries using one or more indexes or repositories generating the results as a ranked list of documents. Indrid waits for connections from runquery and processes queries from network requests. We only created indexes and ran queries.

The indexing process in Indri creates the following data structures for the collection of documents (corpus).

- A compressed inverted file for the collection including term position information.

- Compressed inverted extent lists for each field indexed in the collectionn.
- A vector for each document, which includes term position information and field position information.
- A random-access compressed version of the corpus text.

The document vectors are compressed arrays of numbers, where each number corresponds to some term in the collection.

The Indri query language is based on the Inquery [3]. It can handle simple queries as well as complex queries. It allows phrase matching, synonyms, weighted expressions, Boolean filtering, numeric fields, proximity operators. We used Indri because it is available for research purposes, the source code code and accompanying documentation are updated frequently, and it has been successfully tested in other TREC tasks.

2. Enterprise Track

The goal of the Enterprise Track is to study the issues that arise when searching the documents of an enterprise such as web pages, news or email archives. The two main tasks were Email search and Expert search. The Email search included two subtasks "known item search" and "discussion search". We participated in the "known item search subtask, which consisted on trying to find an email message known to exist. Each of the participants contributed a set of ten query-message pairs of their choice. The selection was performed using the Glasgow search system to find conference announcements or minutes of a meeting containing specific information about a topic or a date. The coordinators of the track then chose 150 query-message pairs to be used as the topics, the first 25 topics for the training set and the rest for the test set. The W3C corpus, an enterprise collection built by the coordinators in 2004 was made available for the track, but the known-item search was based on just the "lists" portion of the corpus.

2.1 Runs

We submitted two official runs. A description of the work done follows.

2.1.1 Building Indexes

We used the Indri search engine without any additional pre-processing or modifications to the source code to build the index. The index was created using the w3c corpus lists with the following parameters:

- class = trecweb,
- metadatafield = docno,
- field = body,
- field = title,
- field = p,
- field = heading

Various indexes were created using different parameters until the results from running queries against the training set produced good results in terms of recall and precision. Although the search engine allows for stemming, Porter or Krovetz algorithm, and for stopword removal we opted to exclude both.

One of the problems that we found when running the queries is related to the inclusion of punctuation and special characters e.g., "”', "/", "?", ")", "#" which generate a syntax error because some of these characters are part of the Inquery query language. We also assume that the parser ignores punctuation and when building the index a term such as "C#" is stored as "C". Thus, it becomes quite difficult to deal with a topic like "HTML tidy in C#".

2.1.2 Running Queries

All the queries were automatically constructed using the only field for the topic, the title. In the first run we created for each topic an Indri query Q of the form $\#combine(t_1...t_n)$, where $t_1...t_n$ are the terms in the topic title. For instance the topic

```
<top>
<num> Number: KI64 </num>
<title> Why doesn't Amaya have a hand-shaped cursor? </title>
</top>
```

translates to

```
<query>#combine(Why doesnt Amaya have a hand-shaped cursor)</query>
```

No document was returned for this query. In this case we assume that punctuation has no influence on the outcome. A topic where special characters are part of the semantics and could not be used to create an Indri query is the one shown below.

```
<top>
<num> Number: KI117 </num>
<title> HTML tidy in C# </title>
</top>
```

had to be created as

```
<query>#combine(HTML tidy in C)</query>
```

The document returned was ranked at position 72.

For the second run we added operators to 58 out of the 125 test queries to improve the results. The operators included were synonyms $\#syn$ and proximity operators $\#n$. For instance topic $KI33$

```
<top>
<num> Number: KI33 </num>
<title> WSD WG f2f ws reference examples </title>
</top>
```

translated to

```
<query>#combine(#1(WSD WG f2f) ws reference examples)</query>
```

The proximity operator #1 means that relevant documents will contain the phrase *WSD WG f2f*, that is we are looking for an exact match. The first run ranked the returned document at 4 and was ranked 1 by the second run. In this case proximity operators did improve effectiveness. However, results for other topics showed that the addition of synonyms and proximity operators did not improve the ranking.

2.2 Results

Results for both runs are presented in Table 1. We show only the topics for which adding operators resulted in ranking improvement.

Topic Tc	Rank First Run	Rank Second Run	Topic	Rank First Run	Rank Second Run
33	4	1	115	25	12
49	2	1	116	no doc ret	1
51	3	2	117	72	1
64	no doc ret	1	119	2	1
70	21	7	122	no doc ret	1
87	2	1	125	no doc ret	4
89	84	12	127	no doc ret	1
91	42	1	133	9	1
101	no doc ret	1	136	no doc ret	3
106	4	1	137	no doc ret	4
107	no doc ret	1	141	67	3
110	6	2	142	3	2
111	no doc ret	5	145	3	1
113	3	2	146	no doc ret	1
114	40	1	148	no doc ret	7
			149	no doc ret	1

Table 1. Improving performance using additional operators

A summary of the results for the first and second run, provided by the coordinators of the task, is shown in Table 2.

Run	Avg reciprocal rank over 127 topics	# Topics target found in top 10	# Topics no target was found
First	0.362	74 (58.3%)	35 (27.6%)
Second	0.502	89 (70.1%)	28 (22.0%)

Table 2. Summary of results

At present, we do not have information for comparison purposes with other systems.

2.3 Discussion

Analysis of the results show that 31 out of 58 queries (55.17%) with additional operators improved the ranking of a document or retrieved a document that was not retrieved in the first run. We observed that using exact phrases worked better than synonyms. We are planning to rebuild the index with stemming and stopwords to ran the same queries; this will help us to determine how much weight stemming and stopwords have in this task.

3. Genomics Track

The goal of the Genomics Track is to provide a framework for evaluation of information retrieval systems in the genomics domain [4]. There were two main tasks namely, ad hoc retrieval and text categorization.

For the ad hoc retrieval task, the task coordinators developed a set of five generic topic templates (GTT), each containing 10 instances (topics), for a total of 50 topics. The five GTTs considered were the following: 1) standard methods or protocols to perform some sort of experiment or procedure; 2) role of a gene involved in a given disease; 3) role of a gene in a specific biological process; 4) interactions between 2 or more genes in the function of an organ or in a disease; and 5) mutations of a given gene and its biological impact. The document collection was the same 10-year subset of the MEDLINE bibliographic database of the biomedical literature. The total of records in the subset is 4,591,008. We used the XML formatted documents distributed in five gzipped files.

The categorization task was a document triage task. The goal of the task was to see how well systems can categorize documents into four categories: 1) tumor biology; 2) embryologic gene expression; 3) alleles of mutant phenotypes; and 4) GO annotation. The data used is from the Mouse Genome Informatics (MGI) system [5], [6]. The documents are a subset of journal articles from the *Journal of Biological Chemistry* (JBC), *Journal of Cell Biology* (JCB), and *Proceedings of the National Academy of Science* (PNAS), for a total of 10196,6043. We worked with XML formatted documents and we did not use any of the additional resources available. The subtask we focused on was *tumor biology.*

3.1 Runs

We submitted two official runs for each task, ad hoc retrieval task and tumor biology categorization subtask. A description of the work done follows.

3.1.1 Building Indexes

We created five indexes, one for each of the XML-formatted MEDLINE files, because we did not have enough disk space to store simultaneously the five uncompressed files. Documents were not pre-processed and we did not modify or add code to the Indri system. The five indexes were created separately. The parameters used were the same for all the files and included the following:

- class = trectext,
- field = title,
- field = text

We used *trectext* as the class because the *xml* class required one document per file. This may be one of the reasons why there were no documents processed for the field $h1$.

3.1.2 Running Queries

All queries were automatically constructed using the file containing the topic as a narrative. For the first run we created an Indri query as we did for the Enterprise Track. Some of the topics included punctuation that we had to remove.For instance the topics 109 and 148 had a "”' and a "+".

```
<109>Describe the procedure or methods for fluorogenic
5'-nuclease assay.
<148>Provide information about Mutation of familial
hemiplegic migraine type 1 (FHM1) and its/their neuronal
Ca2+ influx in hippocampal neurons.
```

For the second run we included pseudo-relevance feedback parameters. the parameter *fbDocs* is an integer for specifying the number of documents to use for feedback. We arbitrarily chose 100.

Although smoothing rules are part of the Indri *runquery* application we included none.

3.2 Results and Discussion

A summary of the results for both runs is presented in Table 3.

Run	Total rel	Total rel_ret	Map	R-prec
First	4584	2825	0.1803	0.2174
Second	4584	2767	0.1642	0.1931

Table 3. Summary of results

The results show that pseudo-relevance feedback did not improve effectiveness, but rather decreased. However we cannot reach a definite conclusion because we ran only one experiment with the first 100 documents. Comparison of our results against those calculated for all the participants running systems automatically show that for most of the queries our MAP was below the median.

One of the problems that we found was that for some topics Indri generated duplicate results at different rankings. To solve the problem we retrieved more than 1000 documents. Thus, after deleting duplicates the number of documents retrieved was close to 1000 and fewer adjustments were necessary.

4. Conclusion

Indri provides a robust framework for research. Experiments for the Enterprise track generated better results than for the Genomics track. Using proximity operators in the Enterprise track improved results, using pseudo-relevance feedback in the Genomics track did not. However, a definite conclusion cannot be reached about the latter without further experimentation. The next step is to test some other capabilities that the system offers and start to modify some of the code to improve our results. Pre-processing documents is another area we plan to explore.

References

[1] Allan J., Callan, J., Collins-Thompson, K., Croft, B., Feng, F., Fisher, D., Lafferty, J., Larkey, L., Metzler, D., Truong, T. N., Ogilvie, P., Si, L., Strohman, Turtle, H., Yau, L., and Zhai, C.: The lemur toolkit for language modeling and information retrieval. *http://www.cs.cmu.edu/˜lemur/*,2005.

[2] Abdul-Jaleel, N., Allan, J., Croft, B., Diaz, F., Larkey, L., Li, X., Metzler, D., Smucker, M., Strohman, T., Turtle, H., and Wade, C.: UMass at TREC 2004: Notebook. *TREC 2004*, 2004.

[3] Callan, J., Croft, B., and Harding, S.: The INQUERY retrieval system. In *DEXA-92*, pp. 78-83.

[4] Genomics Track Web Site: *http://ir.ohsu.edu/genomics*, 2005.

[5] Eppig, J., Bult C., Kadin, J., Richardson, J., Blake, J., and the members of the Mouse Genome Database Group : The Mouse Genome Database (MGD): from genes to mice–acommunity resource for mouse biology. *Nucleic Acids Res* 33:D471-D475. 2005.

[6] Nf, D., Krupke, D., Sundberg, J., Eppig, J., Bult, C. : The Mouse Tumor Biology database: a public resource for cancer genetics and pathology of the mouse. *Cancer Res* 62(5):1235-40. 2002.