Information Retrieval

Lab Assignment 1

Paris Mavromoustakos
Partick De Kok

November 11, 2012

1 Introduction

In this first lab assignment, we had to cope with both indri 5.3 and trec_eval applications' environment and use them to index given sets of documents, apply certain queries and evaluate the results generated.

We Installed indri 5.3 and trec_eval on Mac OS X version 10.7.5 using the Terminal application, which we further used to run all the commands needed for this assignment.

2 Indexing

First of all, after downloading both sets of documents (LAT & GH95) from the link given in Blackboard, we unzipped the document files contained in both folders and added them all together in one single folder. This was the folder to be indexed, using IndriBuildIndex found in folder /buildindex of the indri installation directory. In order to index this folder, we first needed to create the "parameters.xml" file which defines the details of the indexing procedure. The parameters.xml file looks like this:

```
< parameters >
< index > index_outputDIR < /index >
< corpus >
< class > trectext < /class >
< path > index_inputDIR < /path >
< /corpus >
< stopper >
< word > word1 < /word >
< word > word2 < /word >
< word > word3 word4 < /word >
< /stopper >
< stemmer >
< name > stemmer_name < /name >
< /stemmer >
< /parameters >
```

Where, *index* is the directory where the indexing results will be saved, *class* defines how the documents will be processed (they will be considered as ”trectext” documents in this example), *path* defines the folder containing the documents to be indexed, *stopper* incudes possible stopwords (for example, word1, . . . ,word4) and *stemmer* defines the stemming method. It is essential to point out that *stopper* and *stemmer* are not obligatory, and they can be excluded from the document.

To start indexing the documents, all we needed to do was run *IndriBuildIndex parameters.xml* in the Terminal. For this command to run properly, the present working directory should be /buildindex and the parameters.xml file should be stored in that directory aswell. While running, the terminal will print out the status of the indexing process, including the time it takes to run the process and the number of documents indexed. Indexing both sets of documents (LAT & GH95) took us 1:26 minutes and created 169477 indexed documents.

3 Checking the indexing results

In order to check our indexing results, we ran the command *dumpindex index_resultsDIR s* while working in the indri installation directory. The *index_resultsDIR* is already known from the previous step (included in the parameters.xml file) and the parameter *s* at the end of the command represents ”status”, meaning that it will print general info regarding the indeing results. The output we got for indexing both sets of documents without using stopwords or stemming, is the following:

```
Repository statistics:
documents: 169477
unique terms: 335273
total terms: 88270885
```

However, if we run the command using *v* instead of *s*, we would get a comlpete ”vocabulary” table of all unique terms, which would look like this:

```
the 5071448 164890
to 2249861 159030
of 2210672 161030
a 2136668 160832
```

where the first column represents a unique term, the second column represents that term’s total number of appearances in the set of documents and the third column represents the number of documents in which that term was found. In the table above, we see the 4 most ”popular” unique terms in our index.

4 Applying Queries

After having indexed all the needed documents, the next step was to apply given queries on them. To run the queries, the present working directory should be /runquery and the command needed is $IndriRunQuery\ indriqueries.xml\ > output.txt$.

The *indriqueries.xml* file contains significant parameters such as $< index >$, $< count >$ and $< trecFormat >$. The index parameter defines the directory where the indexed documents are saved, count defines the maximum number of results that will be taken into consideration (for example, if count is set to 100 we will consider only the 100 most relevant documents for our queries) and trecFormat should be set to "1" so that indri returns results in trec format.

Moreover, the *indriqueries.xml* file contains the queries that will be applied in the following format:

$< query >$
$< number > C301 < /number >$
$< text > Nestle\ Brands < /text >$
$< /query >$

After running this command (only took a couple of seconds), the file "output.txt" was created, containing the results of each single query applied. The output file contains *count* number of results for each query, defined by its *number*. Every line in this file contains the results of a specific query applied on a document, while the results are sorted by ascending score value and look like this:

C301 Q0 LA102994-0117 1 -5.6513 indri

C301 is the query ID, LA102994-0117 is the document where query C301 was applied, 1 is the ranking of the score, which is represented by -5.6513. The score variable is of high importance because it actually tells us the grade of relevance between the query and the document.

5 Relevance Evaluation

The last step of this experiment was to evaluate the relevance of given queries to the indexed documents. In order to do that, we used the trec_eval script and the qrels.txt file. The qrels.txt file defines if there is any relevance between q query applied and a certain document, and each line looks like this:

C301 0 GH951120-000066 BOOL

Where C301 defines the query ID, GH951120-000066 represents the document ID and the BOOL variable is "0" when the the document is irrelevant to the query and "1" when it is relevant. Important notice: we had to modify the qrels.txt file and erase half of its content because it contained relevance assessments for 50 queries while our indiqueries.xml file only contained 25 queries. As a consequence, running the following command gave us critically false results.

In order to run the relevance evaluation script, while working in the trec_eval installation directory, we ran the command: $./trec\_eval\ -q\ -c\ qrels.txt\ output.txt\ > evaluationResults.txt$ where $-q$ indicates we request results for each query individually (not only general results) and $-c$ indicates that the irrelevant documents (score equal to 0.0000) will also be taken into consideration when calculating the relevance score. After running this command, the file *evaluationResults.txt* was created, containing the output of the relevance evaluation.

indexing method	**unique terms**	**num_rel_rec**	**MAP**	**P5**	**P100**	**P1000**
no stemming, no stopwords	335273	300	0.2131	0.4000	0.1200	0.0120
no stemming, 3 stopwords	335270	301	0.2131	0.4000	0.1204	0.0120
no stemming, 123 stopwords	335150	302	0.2131	0.4080	0.1208	0.0121
porter stemming, no stopwords	245295	398	0.2369	0.3920	0.1592	0.0159
porter stemming, 3 stopwords	245294	382	0.2250	0.3600	0.1528	0.0153
porter stemming, 123 stopwords	245278	381	0.2252	0.3520	0.1524	0.0152

Table 1: Results of trec_eval with different input parameters.

On Table 1, we see the various results after running trec_eval with different indexing and/or stemming parameters. Note that the results shown are calculated over all queries. Num_rel_rec represents the total number of relevant documents retrieved over all queries, MAP stands for "mean average precision" and P5, P100 & P1000 represent precision after 5, 100 & 1000 documents retrieved. Whenever 3 stopwords were used, those were "a", "the" and "of", derived from the results of section 3. The 123 stopwords used derive from the following website: http://www.ranks.nl/resources/stopwords.html

6 Conclusion

After installing and running indri and trec_eval we were able to fully understand their function and behavior. We tried experimenting with the parametrization of the indexing parameters, adding stopwords and stemming methods, and comparing the results we got. We believe that after completing this first assignment we are now ready to face more complicated and challenging tasks.