Lab 2 Test collections

Information Retrieval, 2020

Introduction

Goal The objective of this lab is for you to get acquainted with an IR test collection and the Lemur Indri retrieval system. The English Ad hoc collections are the most prevalent and widely used evaluation collections. We will work with one Ad hoc document collection. The typical genre of the Ad hoc document collections historically have been news.

Instructions

For a G grade, follow lab instructions. You are expected to choose 4 topics, 2-3 different queries each and turn in evaluation results in precision and recall plots.

In this lab you will learn:

- about the contents of the test collection,
- how to index a collection of documents using an IR engine,
- how to formulate your queries in the Indri query language,
- as well as experiment and discuss the effectiveness of different queries.

Lab instructions are prepared for the CLEF (Cross-Language Evaluation Forum) AdHoc News Test suite (/local/course/ir/data/adhoc-news), which contains newspaper articles in English and other languages. You can also choose to work with the Swedish MedEval test collection (/local/kurs/ir/data/MedEvalTK) with different types of articles within the medical domain.

While experimenting with large document collections, it is important to keep disk space you are using up in mind. It isn't good to have all your files in your home directory, because that is backed up many times over, and all those copies of all files there can take up significant resources even though many files don't need to be backed up.

Before you start, familiarize yourself with options for UU lingfil environment at:

`https://stp.lingfil.uu.se/lila/disk/elsewhere/`

When picking up a corpus to work with, you should consider to unpack it in one of the directories mentioned on the above web page.

In this lab you will also create index files, which tend to be large and don't have to be backed up. Consider placing them in /tmp or nobackup.

DATA

- Inspect your data: topics and relevance assessments (qrels). Note you can browse the topics with a web browser e.g.

```
https://cl.lingfil.uu.se/~abasirat/ir/adhoc-news/topics/topics.html
/local/course/ir/data/adhoc-news/topics/topics.html
/local/course/ir/data/adhoc-news/topics/
/local/course/ir/data/adhoc-news/assessments/
```

- Find a corresponding collection of documents. You will work with a single corpus and a single task. The following table shows which tasks can be paired with which data. You can treat them as compatible if there is a cross in any of the minor rows (e.g.. bilingual/monolingual) within the major rows (e.g. AdHoc 2004).

```
https://cl.lingfil.uu.se/~abasirat/ir/adhoc-news/data/files/table_adhoc_news_datasets.p
/local/course/ir/data/adhoc-news/data/files/table_adhoc_news_datasets.png
/local/course/ir/data/adhoc-news/data/collections/
```

put the unpacked corpus under /tmp:

```
$ mkdir /tmp/ir
$ cd /tmp/ir
$ tar -xf /local/course/ir/data/adhoc-news/data/collections/English_data/GlasgowHerald95.tg
```

If you prefer using the GUI you would open GlasGowHerald95.tgz from the web browser. Then choose "Open with Archive Manager" (which is the default) Then press [Extract] Important: Then use the file chooser to choose where to extract it; "Other Locations" => Computer => tmp => [Create folder] "ir" (for example) Then press [Extract] again.

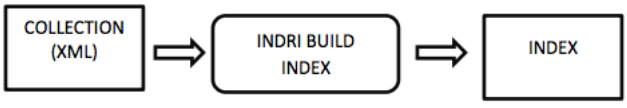
Navigate to your GH95/ directory, it should have your corpus files. This data is your COLLECTION (XML) which we will use for the next step.

INDEX

- Next you will create an index with the Indri toolkit

Build index

The result of this step is INDEX file. In order to build index you need to specify our own index parameter file my_index_parameters.xml. You will feed this file to IndriBuildIndex.



Parameter file for IndriBuildIndex is a well-formed XML document that must be wrapped in <parameter> </parameter> tags. The way how you will use of your parameter file on the command line is:

```
IndriBuildIndex my_index_parameters.xml
```

An example of how your parameter file for building index could look like:

```
<parameters>
 <index>/tmp/gh95index</index>
 <memory>2G</memory>
 <corpus>
   <path>/tmp/ir/GH95</path>
   <class>trectext</class>
 </corpus>
 <stemmer><name>krovetz</name></stemmer>
 <field>
   <name>p</name>
 </field>
</parameters>
```

Let's look at what each parameter in your parameters file means:

index : path to the Indri Repository to create or to add to. Specified as

```
<index>/path/to/repository</index>
```

corpus : a complex element containing parameters related to a corpus. This element can be specified multiple times. For each corpus parameter, you can specify the following items:

path : The pathname of the file or directory containing documents to index. Specified as

```
<corpus><path>/path/to/file_or_directory</path></corpus>
```

in the parameter file or as

```
-corpus.path=/path/to/file_or_directory
```

on the command line.

class : The FileClassEnviroment of the file or directory containing documents to index. Specified as <corpus><class>trecweb</class></corpus> in the parameter file and as -corpus.class=trecweb on the command line.

There are other parameters for specifying corpus, see reference below for more details.

memory : an integer value specifying the number of bytes to use for the indexing process. The value can include a scaling factor by adding a suffix. Valid values are (case insensitive) K=1000, M=1000000, G=1000000000. So 100M would be equivalent to 100000000. The value should contain only decimal digits and the optional suffix. Specified as <memory>100M</memory> in the parameter file and as -memory 100M on the command line.

field : a complex element specifying the fields to index as data, eg TITLE. This parameter can appear multiple times in a parameter file. If provided on the command line, only the first field specified will be indexed. The subelement name is a required field specifying the field name, specified as <field><name>fieldname</name></field> in the parameter file and as -field.name=fieldname on the command line.

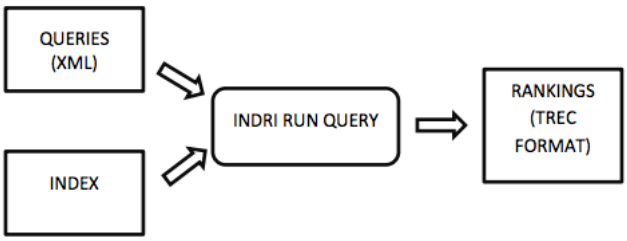
Read more about repository construction parameters on:

```
https://www.lemurproject.org/doxygen/lemur/html/IndriParameters.html
https://sourceforge.net/p/lemur/wiki/IndriBuildIndex%20Parameters
```

Query parameter file

QUERY

- Next you will send in queries – using a query parameter file. You will try out queries with different types of operators: synonyms, ordered window, unordered window, etc.



The result of this step is RANKINGS file (formatted in TREC style). In order to use the IndriRunQuery application you need to specify your own query parameter file my_queries.xml. You will feed this file to IndriRunQuery. Note that your queries should be based on the information needs described as topics in:

```
/local/course/ir/data/adhoc-news/topics/
```

The basic usage of the IndriRunQuery application in command line is:

```
IndriRunQuery my_queries.xml > my_run1_rankings.trec
```

An example of a snippet of my_queries.xml parameter file:

```
<parameters>
<index>/path/to/index</index>
<query>
```

```
<number>1</number>
<text>#combine(silicon valley)</text>
</query>
</parameters>
```

Let's look at what each parameter in your query parameters file means:

query: specifies a query to process. This element can be specified multiple times. This is a complex element consisting of:

number: The query number or identifier. This may be a non-numeric symbol. The default is to number the queries in the parameters in order, starting with 0. This element may appear 0 or 1 times. You should set this to the identifier value of the corresponding information need for which the query is made.

text: The query text, eg, "#combine(query terms)" This element may appear 0 or 1 times and must be used if any of the other parameters are supplied.

type: one of indri, to use the indri query language, or nexi to use the nexi query language. The default is indri. This element may appear 0 or 1 times.

There are other parameters for specifying corpus, see reference for more details.

Read more about query construction parameters on:

```
https://lemur.sourceforge.io/indri/IndriRunQuery.html
https://sourceforge.net/p/lemur/wiki/IndriRunQuery/
```

Read more about the Indri query language:

```
https://www.lemurproject.org/lemur/IndriQueryLanguage.php
http://cs-sys-1.uis.georgetown.edu/~cosc688/wiki/help-qry.html
```

Interpreting the results

The default output from IndriRunQuery will return a list of results, 1 result per line, with 4 columns:

- score: the score of the returned document. An Indri query will always return a negative value for a result.
- docID: the document ID.
- extent_begin: the starting token number of the extent that was retrieved.
- extent_end: the ending token number of the extent that was retrieved

```
-7.45575 GH950114-000120 0 962
-7.83456 GH950814-000122 0 30
-7.9223 GH950504-000131 0 262
-7.99659 GH950302-000002 0 475
-8.02639 GH950930-000171 0 565
```

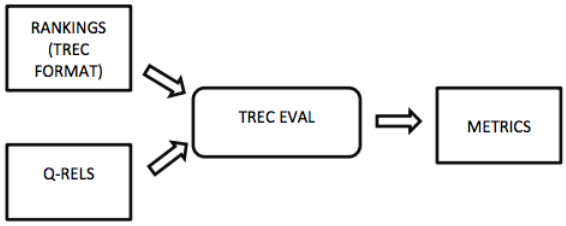
However, we need results in the trec format, therefore add <trecFormat>true</trecFormat> to your query parameter file and rerun IndriRunQuery. The output will be a list of 1 result per line, with 6 columns: <queryID> <iteration> <DocID> <rank> <score> <runID>

```
201 Q0 GH950130-000144 1 -6.64313 indri
201 Q0 GH950823-000149 2 -6.7825 indri
201 Q0 GH950415-000034 3 -7.63488 indri
201 Q0 GH950828-000101 4 -7.70379 indri
201 Q0 GH950519-000000 5 -7.71389 indri
```

Evaluation with trec_eval

EVAL

- Next you will evaluate your results against qrels containing relevance judgements.



Evaluation can be undertaken with a program called trec_eval.

`https://github.com/usnistgov/trec_eval`

It reports average precision at various cut-off points, single value summary measures, and can be used for precision and recall figures (interpolated). Read more about at:

`http://www-nlpir.nist.gov/projects/t01v/trecvid.tools/trec_eval_video/A.README`

For this step you will need two files: RelDocs and DocRank. RelDocs are qrels containing relevance judgements matching queries and documents. DocRank is your result file my_run1_rankings.trec from the previous step. DocRank will be evaluated by comparing it against RelDocs.

RelDocs look like as below and are available in the assessment directory.

```
201 0 GH950102-000036 0
201 0 GH950102-000120 0
201 0 GH950103-000082 0
201 0 GH950104-000002 0
201 0 GH950104-000097 0
201 0 GH950104-000111 1
201 0 GH950104-000117 0
```

You can run trec_eval in command line:

```
trec_eval -q -m official my_selescted_qrels_file my_run1_rankings.trec
```

./trec_eval test/qrels.test test/results.test

num_q	all	3
num_ret	all	1500
num_rel	all	561
num_rel_ret	all	131
map	all	0.1785
gm_ap	all	0.1051
R-prec	all	0.2174
bpref	all	0.1981
recip_rank	all	0.4064
ircl_prn.0.00	all	0.4665
ircl_prn.0.10	all	0.3884
ircl_prn.0.20	all	0.3186
ircl_prn.0.30	all	0.2732
ircl_prn.0.40	all	0.2666
ircl_prn.0.50	all	0.2184
ircl_prn.0.60	all	0.0822
ircl_prn.0.70	all	0.0348
ircl_prn.0.80	all	0.0312
ircl_prn.0.90	all	0.0312
ircl_prn.1.00	all	0.0312
P5	all	0.2667
P10	all	0.3000
P15	all	0.3111
P20	all	0.3667
P30	all	0.3333
P100	all	0.2467
P200	all	0.1600
P500	all	0.0873
P1000	all	0.0437

P = 131/561

Mean Average Precision

Interpolated precision at different values of recall.
Use to draw P/R graphs

Average precision at various counts of retrieved documents

Lab report

You can freely choose 3 topics and prepare 2-3 different queries for each topic. Experiment. Modify queries until you are happy with the returned results. Consider various suitable query language expressions like synonyms, word windows, how specific vs. general your query expression is. All of it will impact how many documents are returned.

For the report you will need to briefly describe which query formulating strategies you found working. Illustrate your report with precision and recall (P/R) graphs. Provide interpretations.

P/R graph can be plotted from your evaluation findings from iprec_at_recall evaluation results. You can use gnuplot function plot. As an input file, you need to create a simple CSV file with x-axis and y-axis data.

VG part

Write a wrapper that automates all the above steps and makes experiment flow easy to reuse with, other corpus, topics, language, etc. Submit your code and instructions how to run it.