Information Processing and Retrieval

Instituto Superior Técnico 2020

Lab 3: IR evaluation

In this lab, we will extend our IR system to be able to evaluate its behavior. In addition, we will make use of existing IR facilities provided by the *Whoosh* search library for Python[1].

1 Querying using Whoosh

The file pri_cfc.txt is composed of several text documents from the CFC collection[2]. It contains one document per line, and the first word of each line is the document ID.

1.1

Program a function that indexes all documents using the Whoosh library. Make sure that you also store the document ID in a separate field. The following code shows an example of how to index documents.

```
import os, os.path
from whoosh import index
from whoosh.fields import *
if not os.path.exists("indexdir"):
    os.mkdir("indexdir")
schema = Schema(id = NUMERIC(stored=True), content=TEXT)
ix = index.createin("indexdir", schema)
writer = ix.writer()
writer.adddocument(id=1, content=u"This is my document!")
writer.adddocument(id=2, content=u"This is the second example.")
writer.adddocument(id=3, content=u"Examples are many.")
writer.commit()
```

Note: The directory indexdir must already exist for the index to be created.

1.2

Implement a function that takes as argument a string and performs a search over the index, returning a list of document IDs, ordered by their similarity ranking. The following example code shows how to perform a query using Whoosh.

[1] https://bitbucket.org/mchaput/whoosh/wiki/Home
All code excerpts shown here are adapted from the Whoosh documentation.

[2] http://www.dcc.ufmg.br/irbook/cfc.html

```
from whoosh.qparser import *
ix = index.opendir("indexdir")
with ix.searcher() as searcher:
    q = QueryParser("content",ix.schema,group=OrGroup).parse(u"first document")
    results = searcher.search(q, limit=100)
    for r in results:
        print (r)
```

Notes: All strings to be processed by Whoosh must be unicode. You can convert any string to unicode using decode method[3] method. By default, Whoosh returns the results ordered using the BM25 similarity.

2 Evaluating IR systems

2.1

Extend your IR system – either the one developed from scratch or using Whoosh facilities – to evaluate answers to queries. To this end, implement **precision**, **recall** and **F1** measures.

These functions should take the following inputs: i) a list of document IDs (the result of a search); and ii) a set with the IDs of the relevant documents.

2.2

The file pri_queries.txt contains a set of queries and, for each query, the set of relevant documents. Program a function that reads the file and, for each query, executes the corresponding search and measures the precision, recall and F1 of the results. The script should print out each query and its evaluation values and a final average value for each measure.

Limit your search results to the first 100.

3 Pen and paper exercise

3.1 Retrieval evaluation

A query Q yielded as answer the ordered set $R = \{d_1, d_2 \cdots d_{20}\}$. In R, documents d_i where i is odd are judged as *relevant*, whereas documents d_i where i is even are judged as *not relevant*. Find the following values:

(a) precision@5;

(b) R-precision;

(c) the interpolated precision and recall curve;

(d) 11-point precision;

(e) BPREF.

[3] http://docs.python.org/2/library/stdtypes.html#str.decode