MERC/IST 2015

Extracção e Análise de Dados na Web

Lab 03: Retrieval Evaluation

To solve the following problems, we will use the *Whoosh* search library for Python.[1]. The goal is to implement and evaluate a simple search engine.

1

The file `aula03_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.

Create a Python script that indexes all the 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.

```
from whoosh.index import create_in
from whoosh.fields import *
schema = Schema(id = NUMERIC(stored=True), content=TEXT)
ix = create_in("indexdir", schema)
writer = ix.writer()
writer.add_document(id=1,
                    content=u"This is the first document we've added!")
writer.add_document(id=2,
                    content=u"The second one is even more interesting!")
writer.commit()
```

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

2

Once the documents are indexed, 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.

```
from whoosh.index import open_dir
from whoosh.qparser import *
ix = open_dir("indexdir")
```

[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`

```
with ix.searcher() as searcher:
    query = QueryParser("content", ix.schema, group=OrGroup).parse(u"first document")
    results = searcher.search(query, limit=100)
    for r in results:
        print r
    print "Number of results:", results.scored_length()
```

Notes:

All strings to be processed by Whoosh must be unicode. You can convert any string to unicode using the `decode`[3] method.

By default, Whoosh returns the results ordered using the BM25 similarity.

3

Implement functions to measure the precision, recall and F1 of a list of search results. The functions should take as input

(a) A list of document IDs (the result of a search);

(b) A set of IDs of the relevant documents

and should return the precision, recall, and F1 value, respectively.

4

The file `aula03_queries.txt` contains a set of queries and, for each query, the set of relevant documents.

Implement a script 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.

You should limit your search results to the first 100.

5

Note: this item is optional and will not be evaluated.

(a) Execute the same evaluation, but using the cosine similarity measure. For instructions, see here: `http://pythonhosted.org/Whoosh/searching.html#scoring`

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

(b) Execute the same evaluation, but removing stopwords. Note that this requires indexing the documents again. For instructions, see here: `http://pythonhosted.org/Whoosh/analysis.html#stop-words`

Which configuration yields the best results?