Web Intelligent Report Lab 2

Master in Intelligence and Interactive Systems

A. Mestre[1] and A. Moral[2]

[1] anagabriela.mestre01@estudiant.upf.edu
[2] albert.lleo3@gmail.com

1 Objectives

The goal of this second lab is in a first instance develop a simple web crawler that collects web pages by following links between those web pages following a BFS approach.

Once we have subtracted that pages, the next step is to develop a complete search engine for those HTML pages. The main purpose is to create an index from a document corpus, just to run queries in batch and interactive mode using those indexes.

2 Methodology

To successfully carry out our task on the web crawler, we will parse the HTML to extract links to other URLs, using a python library called *Beautiful Soup*. In addition, we have used the *urllib.robotparser* library in order to guarantee the restrictions of *robots.txt*: if the page can be crawled and if the crawler should have a delay between crawling the same pages on a website.

For the second part, we subtracted a small subset of web pages with the crawler, and we designed a short list of 5 different queries and a classification to identify the relevant documents for those queries.

After that, we ran the search engine in interactive mode to check if the relevant documents get the best similarities by using *Jsoup* for the parsing and *Porter stemmer* for the stemming

3 Results

3.1 2011-documents results

```
java ti.SearchEngine index 2011-myIndex 2011-documents
```

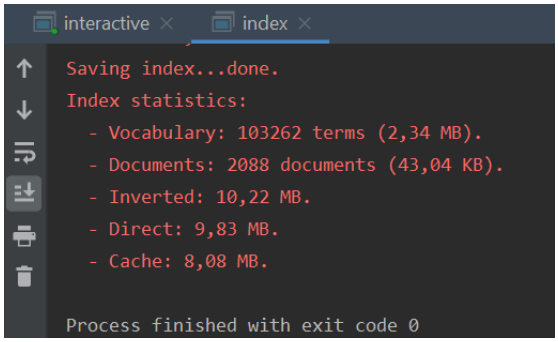

Fig. 1. Index mode

For indexing using the new version of the text processor (HtmlProcessor) we get 103.262 terms instead of the 209.732 terms we obtained with the SimpleProcessor. We have less terms (nearly half the amount of the original version) because we have removed stopwords, words have been normalized (in the SimpleProcessor *Door* and *door* are count as 2 different terms) and they have also been steemed. Therefore, we get a more efficient text processor.

```
java ti.SearchEngine interactive 2011-myIndex
```

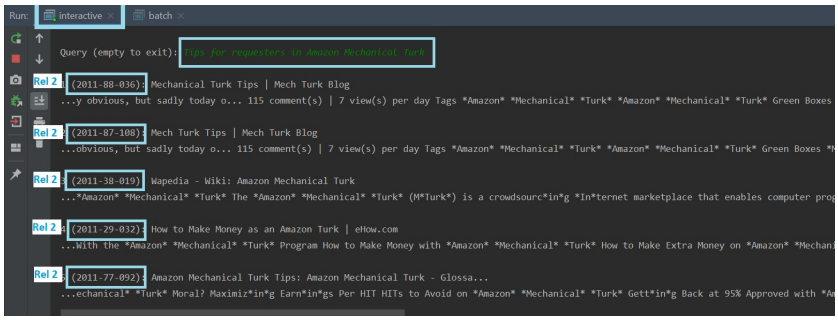

Fig. 2. Interactive mode for query 8

In this example we can see that the first 5 documents that our Search Engine shows have a relevance of 2, which is the highest relevance for that particular query.

```
java ti.SearchEngine batch 2011-myIndex 2011-topics.xml > 2011.run
```

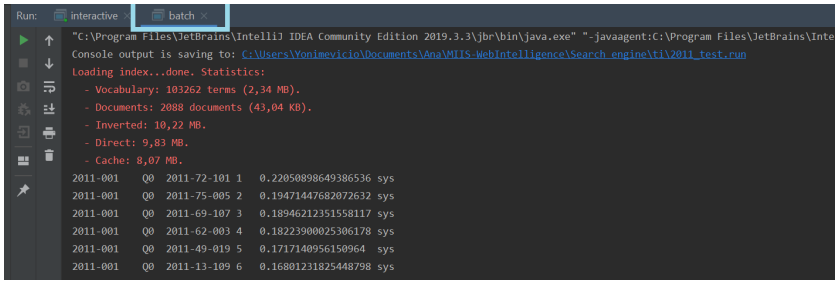

Fig. 3. Batch mode

As for the batch mode, our search engine performs all the queries without any problem and the results are stored in *2011.run* (which can be found attached). This file is used to compare its results with the 2011.qrel. The output for the evaluation tool can be found on the *2011_eval.txt* file. Comparing these results with the ones obtained with the SimpleProcessor, we can see an overall improvement.

3.2 Our own webpages results

```
java ti.SearchEngine index webs-myIndex webs stop-words.txt
```

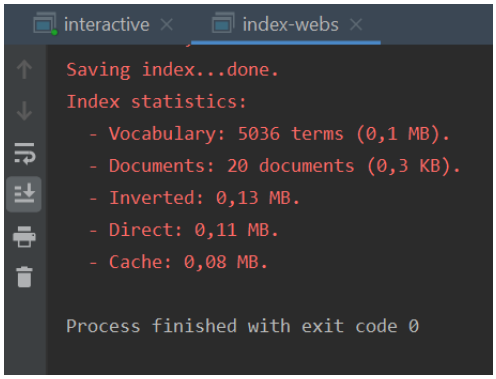

Fig. 4. Index mode

In this case, we have stored 20 webpages. Our index has obtained a total of 5.036 terms. Which obviously is less than before, since in the previous case we were indexing 2088 files.

```
java ti.SearchEngine interactive webs-myIndex
```

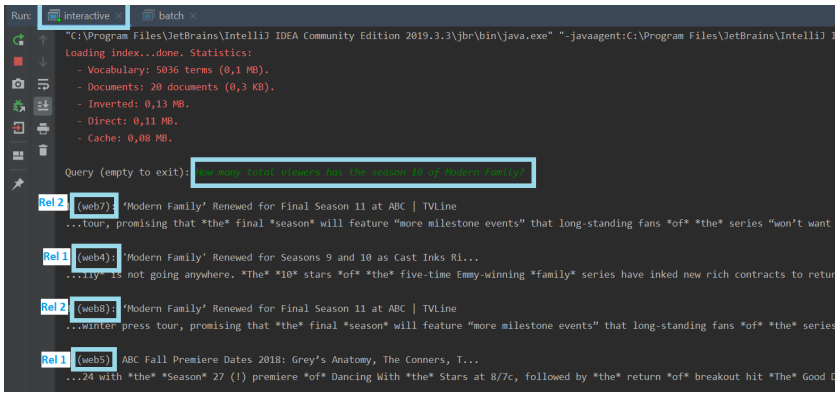

Fig. 5. Interactive mode for query 2

We have tested all our queries and they all have quite good results. However, we have to take into consideration that the interactive mode outputs 10 documents, which made sense before that we had 2088 documents, in this case we only have 20, since 10 is half of it, sometimes not all the top 10 documents must be related to the actual query. Nevertheless, in this case, the first documents that show up have quite good relevance although they don't appear in the order we were expecting.

```
java ti.SearchEngine batch webs-myIndex webs-topics.xml > webs.run
```

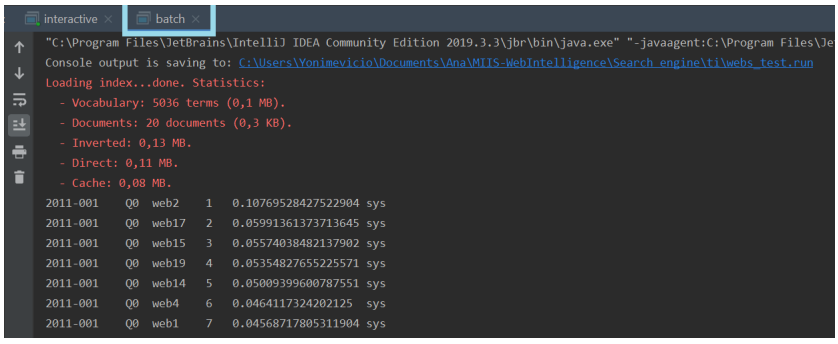

Fig. 6. Batch mode

The batch mode for our own websites is obviously faster than before and stores the results on *webs.run* (which can also be found attached). For this case, we have written a *webs.qrel* with all the relevant documents for each query in the same format as *2011.qrel* in order to run the evaluator in this case too and observe the results. This output can be found on th *webs_eval.txt*.