

Mini Wiki Search Engine

November 6, 2018

We propose here to develop a mini search engine to mine documents from a wikimedia source. A skeleton of the program is provided, and we guide you in the completion of this program to achieve a search engine implementing the standard techniques of Information Retrieval.

1 Setting up your environment

Get the Git repository from the following URL: <https://github.com/ClovisG/WikiSearchEngine> by running the following:

```
git clone https://github.com/ClovisG/WikiSearchEngine.git
```

2 Crawling the data

This section describe how the data has been collected. No coding is asked on your side.

We developed a script called `crawl.py` that requests the Wikipedia API for list of pages in a given category. By looking in the script, you will discover a straightforward querying to <http://en.wikipedia.org/w/api.php> asking for "categorymembers" of a given category. You can choose to retrieve the pages in the subcategories up to a depth controled by the parameter `crawlingDepth`.

After checking (quickly) the code, you should be able to answer the following questions:

Q2.1 Which Wikipedia category is crawled in this script?

Q2.2 What does this script output?

Q2.3 When running the script like `python3 crawl.py > wiki.lst`, what does the file `wiki.lst` contain?

When setting `crawlingDepth = 1`, we get 2096 pages, and when setting it to `crawlingDepth = 4`, we got 69 867 pages. This will provide you two datasets, the first one for debug purposes, and the last one for final testing.

3 Downloading the data

We developed a simple bash script `dw.sh` that takes the file `wiki.lst` as argument and download the related Wikipedia pages. Due to some limitation on the Wikipedia side, we chose to download the pages by batches.

By looking in the script, can you tell:

Q3.1 How many pages per batch is downloaded?

Q3.2 What API of wikipedia is used to download a set of pages?

Q3.3 By going to the API page in your browser, can you tell in what format the pages will be encoded?

4 Parsing the data

Take a look at the files containing the Wikipedia pages in the directory `dws`.

Q4.1 Give a regular expression to extract all the titles of pages encoded in this file.

A parser of the Wikipedia documents has been developed in the `parsexml.py` that create a *tokdoc* matrix as well as a jump matrix for the random surfer model.

TODO detail what is a group TODO remove reg ex for links

Q4.1 From the code, how are encoded the two matrices (*i.e.* what type of Python object)? What is the name of this encoding?

Q4.2 Take a look at the content of a Wikipedia page in the `dws` folder. How are the links encoded in the wiki language?

Q4.3 The regular expression for extracting the links has been removed. Propose a regular expression to detect the links in the Wikipedia format.

Q4.4 Implement your regular expression in Python such that the first group contains the link description.

5 PageRank of the documents

Q5.1 Run the PageRank program in interactive mode: `python3 -i pageRank.py`, and use the Python interface to answer the following:

- What is the page rank of "Charles Darwin"?
- What is the page with the highest rank?

6 Search!

TODO make the search in non-normed mode.

If you run in interactive mode the `search.py`, get the best 15 results by the vector model for the query *theory of evolution*.

Q6.1 What type of page is selected by the vector model? By looking at the Wikipedia page, how can you explain it? What is the name of this classical *cheating*?

Q6.2 Propose and implement a way of correcting this phenomenon. Check if this correct the effect for the top 15 pages.

Q6.3 Rank the results according to pageRank (using the `rankResults` function), and print them using the `printResults` function. Does it look nice?

Q6.4 Take a look at vector model rankings for the query *evolution of bacteria*. What is the rank of the page "*Bacterial evolution*"? Rank the results by pageRank. What is the rank of the page "*Bacterial evolution*"? Is it expected? How would you correct for it?

7 Extras

You can tune the page source in the Random Surfer Model to put more emphasis on the source vector, as well as modifying the source vector itself. Note that handling this properly is a bit counterintuitive.

Q7.1 What is the page rank of the page "DNA"? What is the page rank of the page "RNA"?

Q7.2 For example, set the source vector as the pages with DNA in the title. What is the page rank of "DNA" and "RNA"? How can you explain this? How should you modify the source vector in order to increase the "DNA" page rank?