Task 2. Crawler

Javier García, Jesús Matos, Liam Mahmud, Krish Sadhwani

November 2, 2022

Abstract

Previously, an inverted index was made, which would work on a collection of documents, however, the project would be improved if the process of downloading and storing these collection of documents could be automated.

For this, the crawler module was created, capable of fulfilling this function, a series of experiments with the Project Gutenberg site have been carried out and the results have been conclusive, the crawler is capable of downloading and storing files from this set automatically, for this project, the implementation was carried out to work every minute.

1 Context

The purpose of this project is to create a crawler module, this module has the functionality to download a document every minute and save it to the document repository. The organization of this repository is chronological with folders for each date. All project files as well as tests performed are in the following GitHub repository: Crawler Repository

2 Problem Statement

Initially, a library called Project Gutenberg is considered in which there are approximately 60k books in its collection, in the first work, the idea of an inverted index was constructed to use with this data set, and subsequently, a module that automatically downloads and stores books for further processing.

3 Methodology

Firstly, an inverted index module in java was created, for this the previously based delivery made in Python was considered: Inverted Index Repository

During the implementation of the inverted index in java, the decision to fix several mistakes had been made in comparison with the previous implementation in Python.

More classes were implemented to separate the code and increase its clarity.

In addition, a Document class was made, in which the metadata of the documents extracted with the crawler were stored.

This time, three different languages were considered where a document may have to delete it's stop-words: Spanish, English and French.

These three language stop-words, are stored in three different text documents and are chosen depending on which one to use after checking the document language in its metadata.

Also, the creation of the inverted index with Hashmaps was also utilized, which delivers faster execution.

Lastly, the midpoints were removed (Inverted Index of individual documents), and instead, every time a bunch of documents were indexed, it were to be added in the inverted index or the values replaced if the document was already stored in it, in case it was edited.

Then, the crawler concept was discharged, for this a division of three modules had been performed:

- Crawler Package: Main module, automates the process of downloading and storing the files.
- Downloader: Downloads Project Gutenberg data.
- MetaData Extractor: Extracts metadata from the text file.

4 Experiments

4.1 Tests

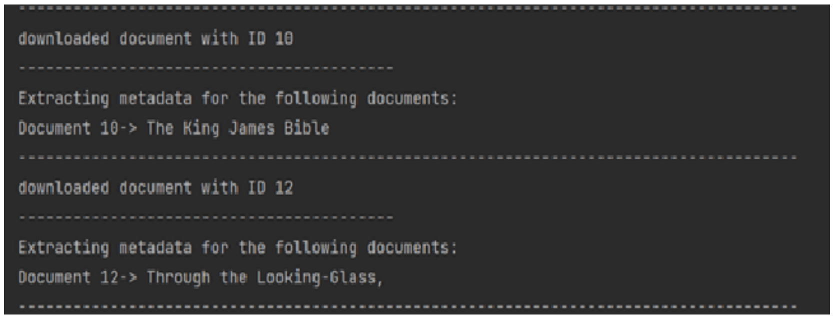
To highlight some issues faced:

With first instances, the download of the files from the Project Gutenberg site was performed randomly and not sequentially. This generated a problem, the time of the execution varied a lot through the iterations, this was due to the fact of the program having to perform the task on deleted documents, those that did not exist. So a final implementation came into play, one where the download would be sequential and the id of the last downloaded document would be saved in a Properties file. This time when the Crawler performs its task again, no repeated documents are downloaded and it avoids general inconsistency.

Another issue to have had confronted was the extraction of the metadata from the files, as there were sometimes more than one line for each field. So the recognition of every parameter had to be implemented in such a way, that it would always get all of the data from all fields in the metadata. Of course, through trial and error, a function was finally constructed that would give the correct output.

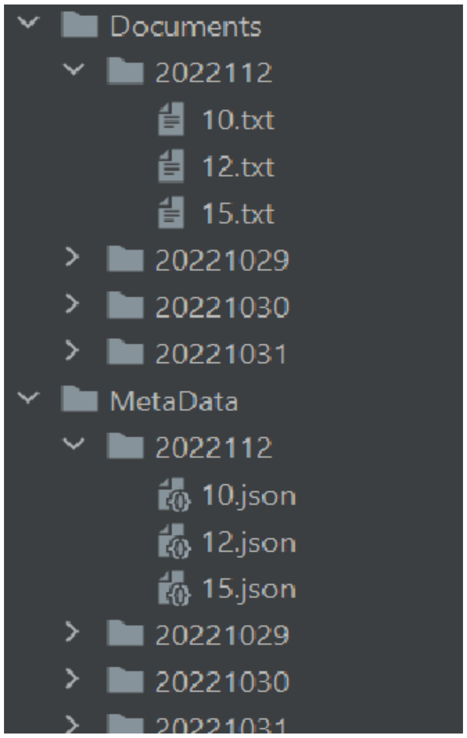
When generating the Json files through the first versions, it stored it in the same directory where the text documents obtained from the Project Gutenberg site where located. This was a problem, as the definition of the task was to segment these files in different folders correctly named and considering placing them through a folder for each dates. This was corrected utilizing different Java packages to read and place files given certain parameters, and manipulating the path constituted to the files, so that everything would be rightly placed.

Below, images of the results obtained from the testing phase are attached. To begin with, messages printed in the console can be observed to verify that the whole process works correctly:



```
--------------------------------------------------------------------------------
downloaded document with ID 10
---------------------------------------
Extracting metadata for the following documents:
Document 10-> The King James Bible
--------------------------------------------------------------------------------
downloaded document with ID 12
---------------------------------------
Extracting metadata for the following documents:
Document 12-> Through the Looking-Glass,
--------------------------------------------------------------------------------
```

Also, one is able to verify that folders are being generated whose name is the date on which it has been created, and in them, both text files and Json files with metadata are stored.



```
Documents
    2022112
        10.txt
        12.txt
        15.txt
    20221029
    20221030
    20221031
MetaData
    2022112
        10.json
        12.json
        15.json
    20221029
    20221030
    20221031
```

5 Conclusion

In summary, the work has consisted itself of the creation of a crawler which downloads and stores files, this crawler has been divided into several modules, one of them automates the process, another downloads the documents and another extracts the metadata. Finally, we have done a series of experiments with the data obtained from the Project Gutenberg site whose results are quite positive, the program is able to download and store the files every minute.

6 Future work

As a tip to improve the work, experiments could be conducted with another data source other than the Project Gutenberg collection.