Task2. Crawler

Javier García, Jesús Matos, Liam Mahmud, Krish Sadhwani

October 2022

1 Abstract

Previously, we have made an inverted index which works on a collection of documents, however, this project could be improved if we managed to automate the process of downloading and storing this collection of documents. For this, we have created the crawler module, capable of fulfilling this function, we have carried out a series of experiments with the Project Gutenberg file collection and the results have been conclusive, the crawler is capable of downloading and storing files from this set automatically, in our case we have implemented it to work every minute.

2 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

3 Problem Statement

Initially, we have a library called Project Gutenberg in which there are about 60000 books in its collection, in our first work we managed to create an inverted index to use with this data set, and this time, we have created a module that automatically downloads and stores books for further processing.

4 Methodology

First, we have created an inverted index module in java, for this we have based on our previous delivery made in python: Inverted Index

When implementing the inverted index in java, we made the decision to fix several mistakes we made when implementing it in python. We first implemented more classes to separate the code and increase its clarity. In addition, we implemented a Document class, in which we store the metadata of the documents extracted with the crawler. This time we took into account 3 different languages that a document may have to delete it's stopwords: Spanish, English and french. These 3 languages' stopwords are stored in 3 different text documents and we choose which one to use after checking the document langauge in its metadata. Also, we implemented the creation of the inverted index with hashmaps, which makes it much faster. Lastly we deleted the midpoints creation (Inverted index of individual documents), and instead, every time we index a bunch of documents, we add them to the inverted index or replace their values if the document was already stored in it, in case the document was edited.

Then, we have made the crawler, for this we have divided it into three modules:

Crawler Package: Main module, automates the process of downloading and storing the files.

Downloader: Download Project Gutenberg data.

MetaData Extractor: Extracts metadata from the text file.

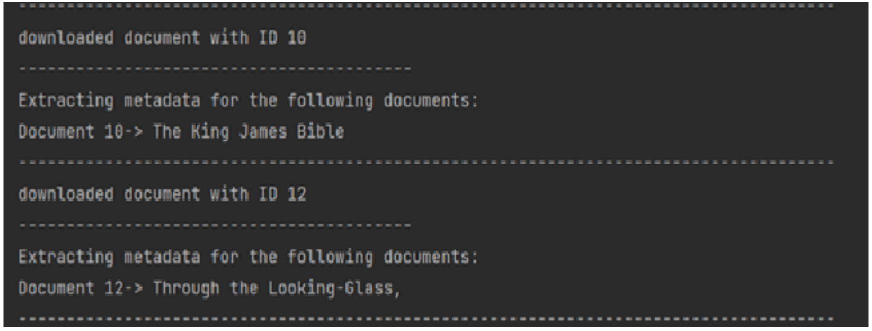
In addition, the downloads are done sequentially but we save in a .properties file the id of the last downloaded document so that when we run the crawler again, no repeated documents are downloaded.

5 Experiments

5.1 Tests

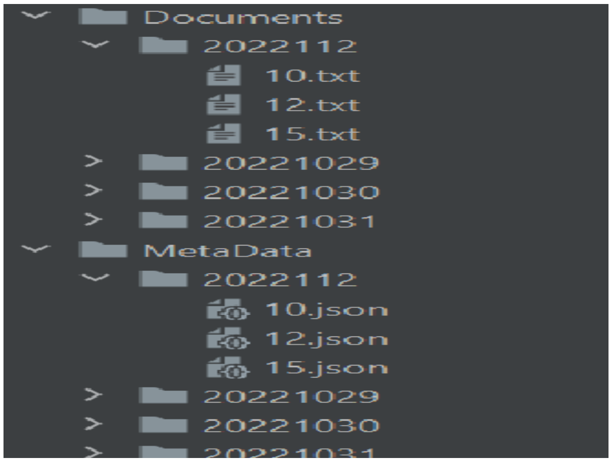
Below, we attach images of the results we have obtained from the testing phase:

To begin with, we observe how messages are printed in the console 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, we ourselves are 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
```

5.2

6 Conclusion

In summary, our work has consisted 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 Project Gutenberg data collection whose results are quite positive, the program is able to download and store the files every minute.

7 Future work

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