Task 1. Inverted Index

Jesús Matos, Javier García, Liam Mahmud, Krish Sadhwani

October 10, 2022

Abstract

Inverted Index functionality for documents processing.

1 Context

The goal of this task is to formulate a program that would build a word-level inverted index function taking as parameters a collection of documents. The reason why such proposition is made is due to the fact that the approach applies a rapid and efficient search of information and it is vital for major corporations like Google, Meta, Apple..etc to utilize this technique as they deal with large quantities of data.

All of the project files, as well as the test cases performed, are available at this very github repository link:

Inverted Index Project

2 Problem statement

A first initial idea is to develop a function that would take files in a designated folder, process, and analyze it's content to then further create Json files for each document that would contain the indices of the words in a dictionary format.

Apart from this, if the files from the folder are internally modified, the program should also be able to update the IE from the Json files as well.

3 Methodology

The structure is emphasized to have three classes, one that would invert the index, the other will be in charge of verifying the documents to be processed and finally another that would normalize the contents of the files for word detection.

Everything will be called from the main function where one could also alter the main path, as the folder to be accessed is the "Documents" folder where the search will take place from the system.

All of this is to be programmed in the Python language, which provides the facility of user-readability and a wide range of libraries.

4 Experiments

4.1 Versions and parallel ideas

One of the parallel ideas of this project was based on the creation of a code that generates a file for each word of the document to be indexed. Obviously, this first version was discarded because of the slow performance and the weight of countless files. However, thanks to this one, the chosen one emerged and a previous step was found that can be very useful depending on the amount of information one is going to handle. An so, a new method would have to be utilized, that would check if the entered document already existed in the files and if it was not, the user would be shown that a single Json object was to be created for the document, and then it would be added to the inverted index. This implementation has been called the **"midpoint"**.

4.2 Simple use case

To demonstrate a simple use case for testing purposes of this program built, a text document with a few lines was created to be later processed, within the "Documents" folder of the system:

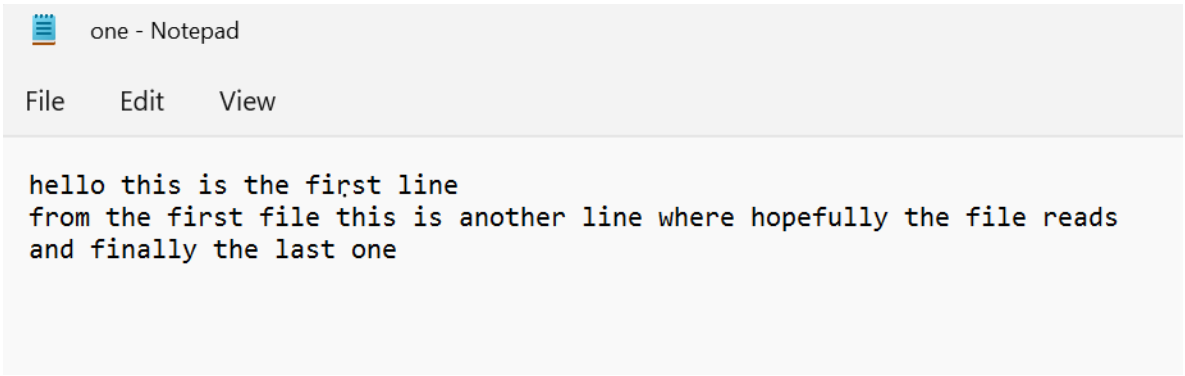

Figure 1: Screenshot of the file's content from notepad.

The Json file created after running the program is such, where the values from the dictionary format demonstrate the indices in which the words are located:

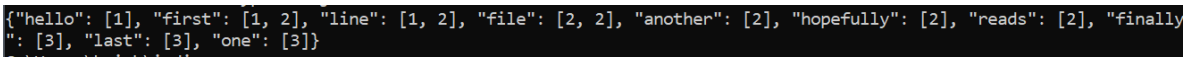

Figure 2: Screenshot of the Json's content from the Windows terminal with "type" command

This proves it's validity.

4.3 Checks and possible problems

Apart from this tiny experiment, large files have also been tested, this is available in the Github repository attached previously, within the "Documents" folder. Different samples have been used for testing, where the output of Json can be seen in "IE.json" file. Which in turn, due to the sheer length and size, can be viewed upon in a raw format.

By testing different files and documents, possible bugs were noticed in the code, for example when reading certain ASCII characters the program reacted by saving the word in a distorted form and not in its correct form. On the other hand, no problems were encountered with the execution of large documents.

4.4 Problem Solution

For the problem with the ASCII characters, a function was found where if the following parameter was used; "ensure_ascii" as false, the problem was solved.

5 Conclusion

As it had been said before 4.1, thanks to the way of designing the code, a **midpoint** was managed to be created, a previous step to the inverted index, which had its main purpose be revolved around the fact of verifying the existence of a given document and performing certain tasks depending on the conditions.

This would seem as one of the most important things of the project because in large companies such as those mentioned above, the amount of information they get from their sources is uncountable, so if what interests them is to search for a series of references in certain documents, they can do it by directly accessing the Json of the said document and search for it.

6 Future work

As an annotation to improve in future work, it would be ideal to be able to control several important parameters for faster execution and debugging of possible errors.

For example, two of these parameters to control would be the identification of the language of the document, for the best possible removal of stop words, and the necessary conversion of these files to UTF-8.