Inverted Index

Andrea Hernando González, Cynthia Quintana Reyes, Aarón Perdomo Aspas

October 11, 2022

Abstract

The following work presents the development of a Python function that creates an inverted word-level index from a collection of documents.
An inverted index is a database index that stores a mapping of content, such as words or numbers, to their locations in a table, or in a document or set of documents. The purpose of an inverted index is to allow fast full-text searches, at the cost of increased processing when a document is added to the database. The inverted file can be the database file itself, rather than its index. It is the most popular data structure used in document retrieval systems, used on a large scale, for example, in search engines.
Our function processes two text files from our GitHub repository, removes special characters, spaces, and stopwords from the text, and returns the documents and line numbers where each word is found.
In conclusion, we have stored the location of each word in a list.

Introduction

The programming language that we have used to develop this work has been Python, which is a high-level interpreted programming language whose philosophy emphasizes the readability of its code, it is also a multi-paradigm programming language, since it supports partially object orientation, imperative programming and, to a lesser extent, functional programming. It is an interpreted, dynamic and cross-platform language.
To develop the code, we have used Pycharm, which is an integrated development environment (IDE) used in computer programming, specifically for the Python programming language.
The inverted index is a topic of relevance that has been studied for a long time by various scientists. Some interesting articles related to this topic are: "Inverted index based multi-keyword public-key searchable encryption with strong privacy guarantee" by the authors: Bing Wang, Wei Song, Wenjing Lou and Thomas Hou, and "Building an Inverted Index at the DBMS Layer for Fast Full Text Search" by the authors: Ciprian-Octavian Truica, Alexandru Boicea, Florin Radulescu.
This document explains how we have developed the different experiments with our inverted index algorithm, which tries to process text files from a Github repository, remove all special characters, stopwords and spaces from said files, and return a dictionary with each word of the texts without repetition, indicating in which document number and in which line number each of them is found.

Methodology

The inverted index is built like a dictionary – a collection of keys and values (Figure 1). The key is about the document in which each word is found, and the value indicates the line number where that word is in that document.

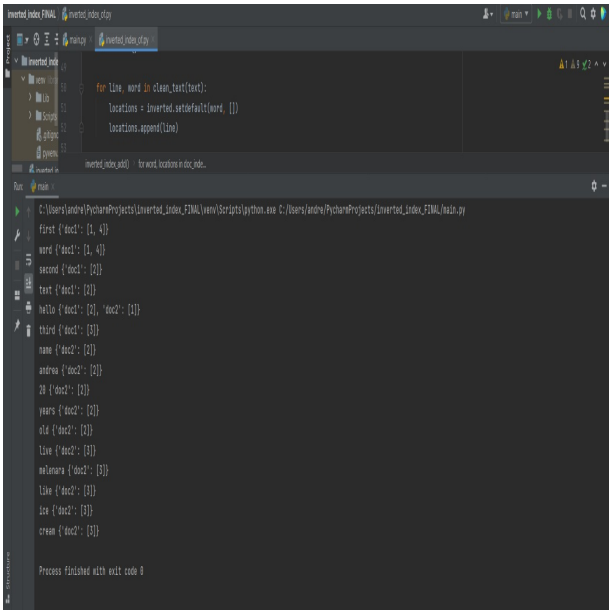


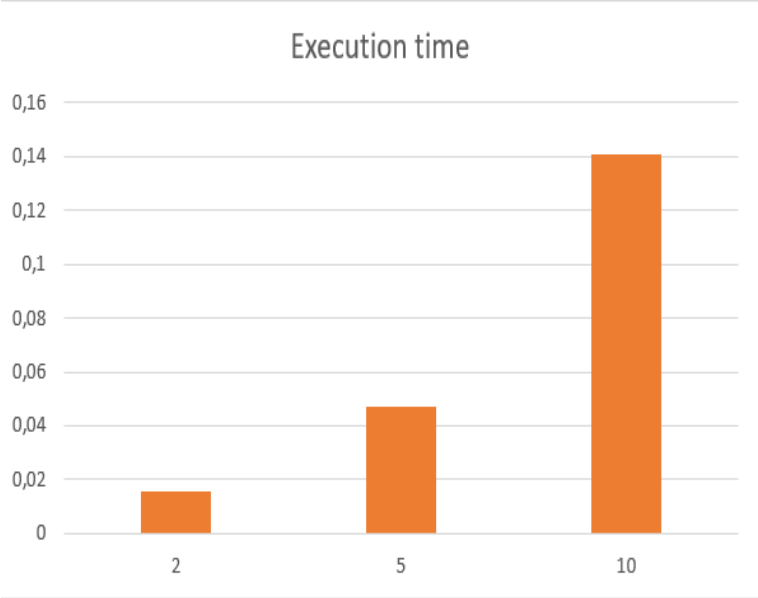
Figure 1

The project is developed in Pycharm and in the Python programming language. Our project consists of two files: invertedindexof and main, respectively, the first file contains seven functions:
The open document function reads text files.
The delete punc split function processes the text by removing special characters, and adding each word of the text to an empty list that we created at the beginning.
The normalize words function normalizes the words in the list, that is, converts all words to lowercase.
The delete stopwords function removes the stopwords from the list.
The clean text function calls delete punc split, normalize words, and delete stopwords, to process the initial text.
The inverted index of and inverted index add functions perform the inverted index operation.
The main file is where we pass the url of the texts uploaded to Github so that the program executes them by calling the functions of the Inverted index of file, and we print the dictionary to show us the result.

Experiments

We have two different versions of inverted index code. The first one is like the first idea we developed and then the last version of the project. We are going to do some test to measure the time execution of each algorithm and why the second one is better than the other.
First let's explain a bit both functions. In the first version we implement only one function that make every changes. But this is not a good practice, we must avoid having only one function for everything, it's better to write many small methods. This helps us to avoid repeating code if we want to implement similar things.

We test these two functions for 2, 5 and 10 documents. Every document has around 40000 words and we obtained the following results Using only two documents we can see that the only one function version start to have some problems and comparing execution time this one last 0.109 seconds while the other one is seven times faster with a result of 0.0155.
In the following image we see the execution time used to do the inverted index for 2, 5 and 10 documents. And we see that, it is able to develop it for 10 documents with more than 50000 words in less than a second.



Conclusion

The inverted index is a relevant technique, as it allows you to store a mapping of content, such as words or numbers, to their locations in a document or set of documents. In simple words, it is a hash map-like data structure that directs you from a word to a document or a web page.
The functions that we have used to develop the project allow us to read text files from a Github repository, eliminate stopwords, special characters, normalize said texts and obtain the inverted index, which results in a dictionary with keys (number of documents) and values (lines of such documents).
In conclusion, this program will allow us to find the locations of the words easily and effectively.