# PROJECT REPORT ON TF-IDF Query Retrieval on Shakespeare’s Corpus

A Project Work Submitted in Partial Fulfilment of the requirements for

The Course

**ALGORITHMS FOR INTELLIGENCE WEB AND INFORMATION RETRIEVAL**

**By**

## Manoj Mahesh Patil PES2201800656 Hrithik Diwakar PES2201800666 Sumukh R.R PES2201800078

Under the supervision of

Prof. Nagegowda K S

Associate Professor

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

PES UNIVERSITY

EC CAMPUS

## TABLE OF CONTENTS:

1. INTRODUCTION

2. OVERVIEW OF THE PROJECT

3. ALGORITHM PSEUDO CODE

4. RESULTS AND DISCUSSION

5. CONCLUSION AND FUTURE SCOPE

6. REFERENCE & SOURCES

**1. Introduction**

Term Frequency is defined as the ratio of the number of times the “Term” appeared in a document to the total number of Terms in the document.

Inverse Document Frequency is the ratio of Total Number of Documents to the Number of Documents Containing the Term.

The product of the Term Frequency and Inverse Document Frequency is TF-IDF. This method is widely used for Information Retrieval and Text Mining. Computer can understand data in the form of numerical values , so for this reason we vectorize all of the text so the computer can understand better. By vectorizing the documents we can perform various numerical operations like ranking , clustering and so on. This is the same thing. This is the same thing that happens when you perform a Google search. The web pages are called Documents and the search text is called Query. When you search with a query, google will find the relevance of the query with all of the documents, ranks them in the order of relevance and shows you the top k documents.

**2. Overview of the Project**

The data that we are using is Shakespeare’s Poems Corpus in a folder called **Stories** . The folder contains 23 Shakespeare’s Poems in .txt format . Each document has different names. There is also an **index.html** file which contains the name and the location where all these poems are stored which can be used for efficient query retrieval.

The project is not only limited to Shakespeare’s Corpus , it can also be applied to any corpus that is in text (i.e. .txt ) format.

**3. Algorithm and Pseudocode**

As mentioned in the title we will be using TF-IDF as well as Cosine similarities to rank documents. In order to rank the documents according to their TF-IDF / Cosine similarities we have to first convert the input string to a set of Tokens viz called as Tokenization.

For this , we follow the below steps :

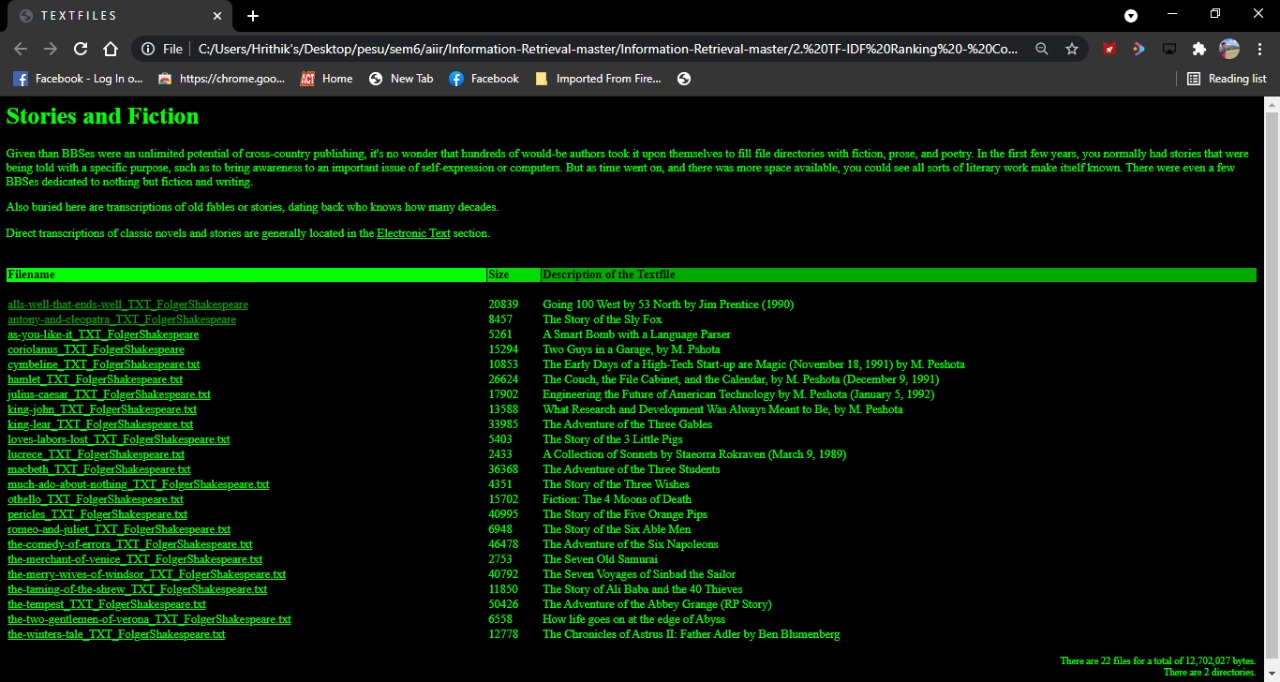
* We convert all words into lower case.
* Stop word removal using **ntlk** library.
* Punctuation removal.
* Elimination of Single Characters.
* Conversion of Numbers to Alphabets (For ex. $100 is converted to hundred dollars).
* Lemmatisation.
* And the most important part of pre-processing , Stemming.

Finally , after doing all this we calculate :

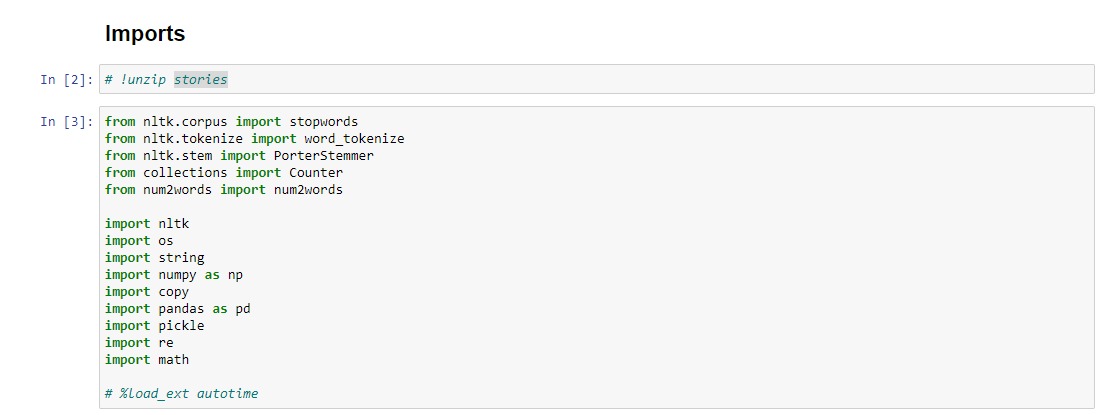
* TF-IDF = Term-Frequency \*Inverse Document Frequency
* Cosine Similarity = Dot Product of vector a and vector b.

Since TF-IDF is a way to measure the importance of tokens in text and Cosine Similarity is used to compare the similarity of the query words with those present in other documents , we use Cosine Similarity as our final ranking parameter.

The **index.html** file:



The libraries that we need :



**Pre-processing :**

**Step 1 :Extracting Data:**

All the corpus is stored in the **stories** zip folder in .txt format . First we need to extract them

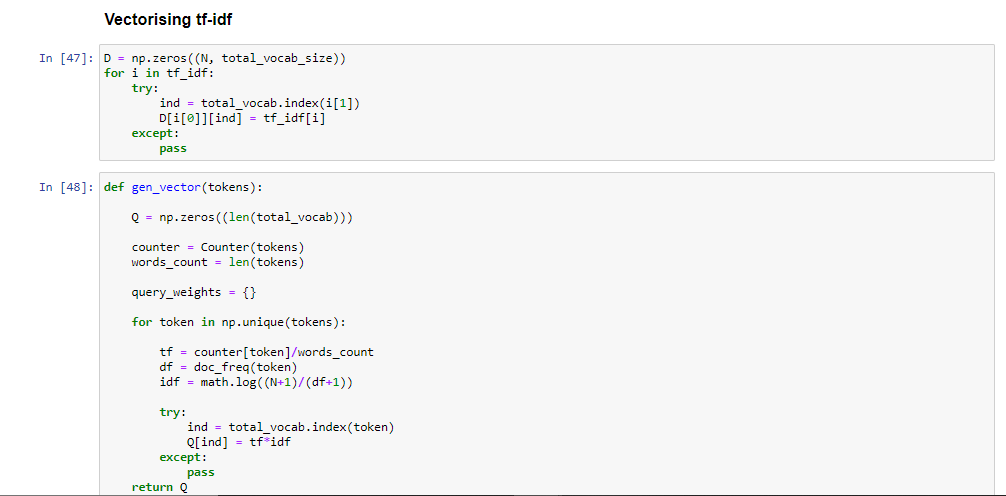


**Step 2 : Collecting File Names and Titles :**

Next , we create an empty list called **dataset**  which contains the titles of all the .txt files in the stories folder.

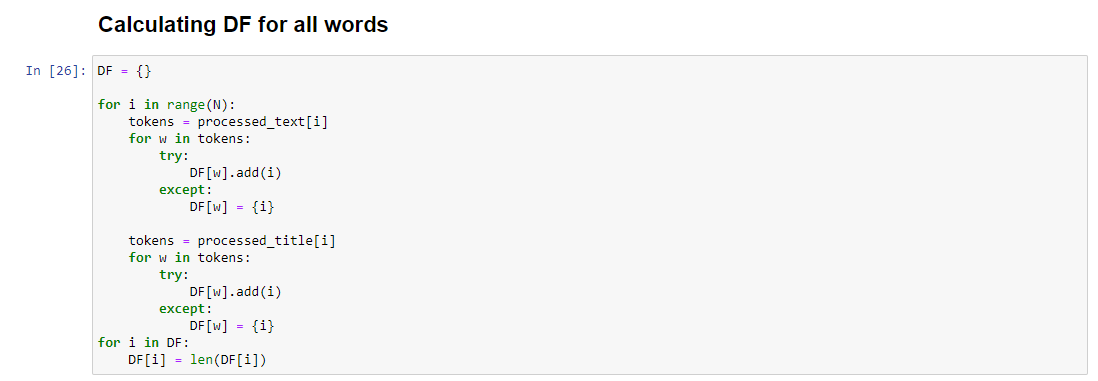


**Step 3 : Converting Strings to Numbers**



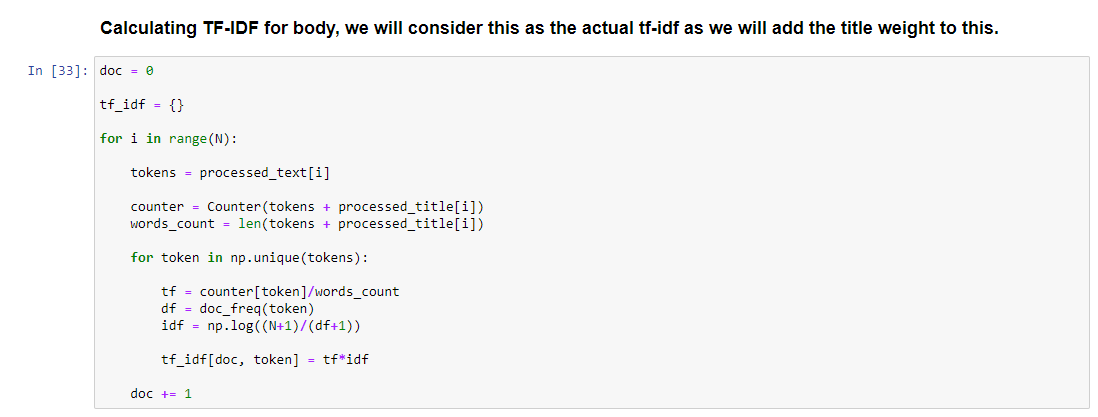
**Step 4 : Calculating Document Frequency**

Document frequency is defined as the ratio of the Number of Documents containing the Term to the Total number of Documents.



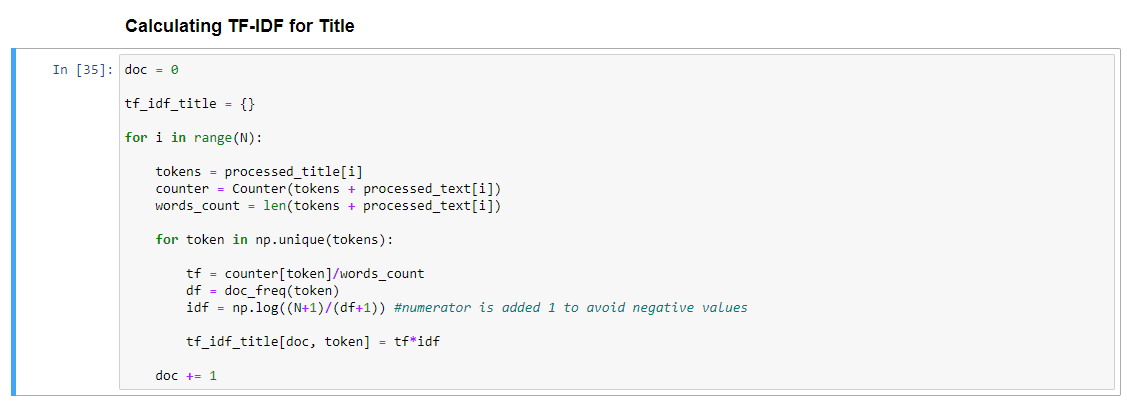
**Step 5 : Calculate TF-IDF of the body.**

We calculate the TF-IDF value to the body of the document , ignoring the Terms in the Title.



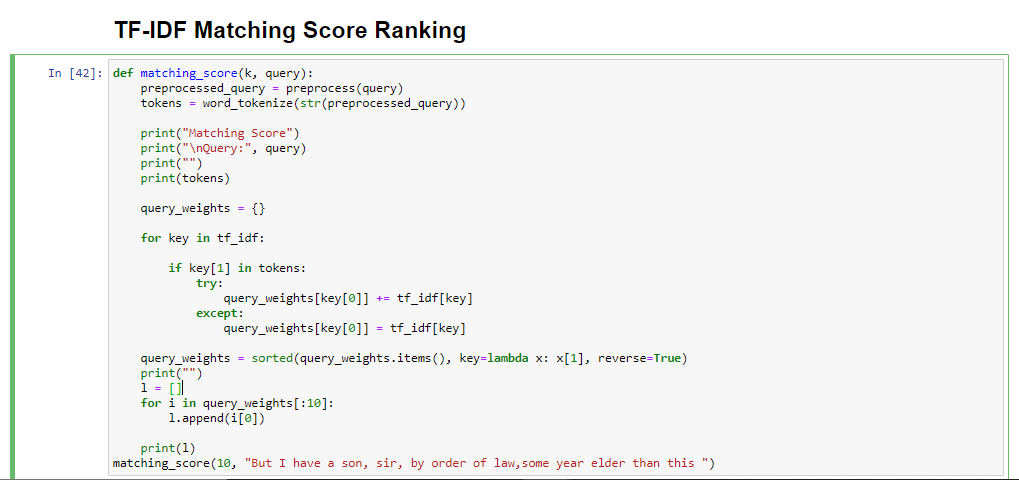
**Step 6 : Calculate TF-IDF of the Title.**

We calculate the TF-IDF value of the Title , ignoring the Terms present in the Body of the document.



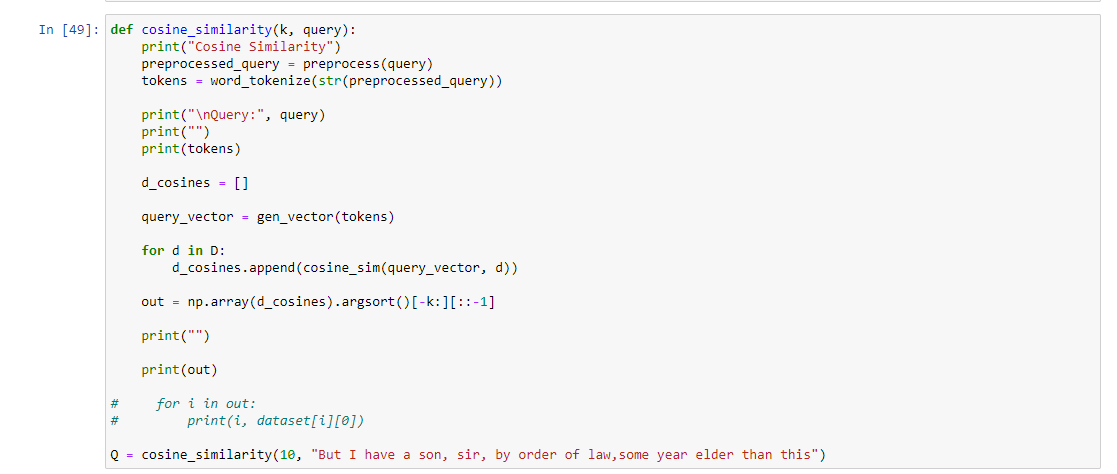
**Step 7 : Calculating TF-IDF Matching Score**

We multiply the weight alpha to the body and 1-alpha to the title and find the Matching Score.



**Step 8 : Calculating Cosine Similarity**

We find cosine similarity using dot product

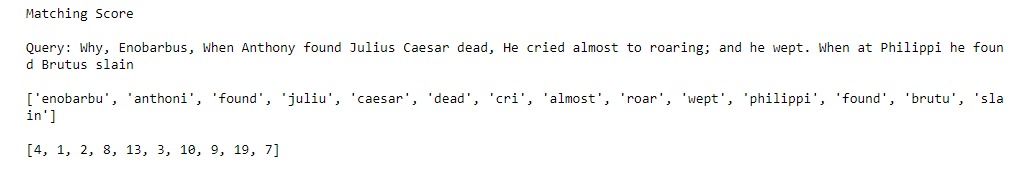


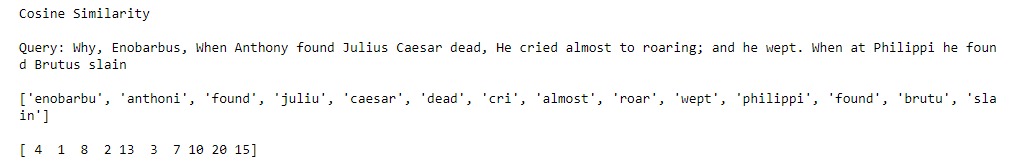
Using the above 2 similarity methods we rank the documents.

**4. Results and Discussions**

The output rankings for the following queries are :

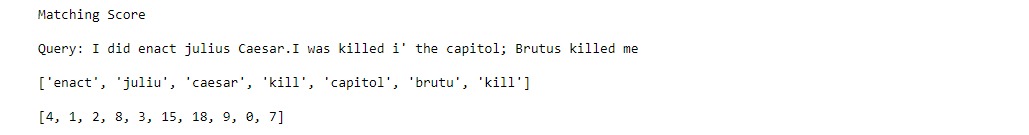
**Query 1 –** “Why , Enobarbus , When Anthony found Julius Caesar dead , He cried almost to roaring ; and he wept . When at Philippi he found Brutus slain ”

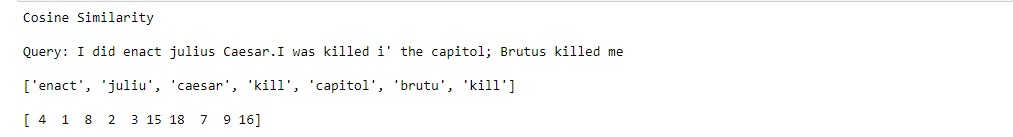




Hence , seeing the above rankings we can say that Document 4 is highly ranked.

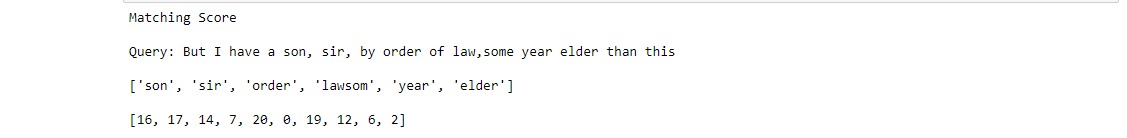
**Query 2 –** “I did enact Julius Caesar . I was killed i’ the capitol ; Brutus killed me”

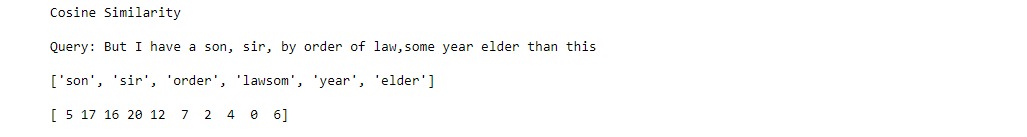




Hence , seeing the above rankings we can say that Document 4 is highly ranked.

**Query 3 –** “But I have a son , sir , by order of law , some year older than this”





Hence , seeing the above rankings we can say that Document 17 is highly ranked.

**5. Conclusion and Future Scope**

As a part od our 6th semester AIW&IR project we have created a TF-IDF Query Retriever on Shakespeare’s’ Corpus.

There is still room for improvement in our Project , for example our Project is only limited to Corpus which is in .txt format , we can expand our project to include more document types and use more complex algorithms to retrieve queries.

**6. References & Sources**

We referred the following websites to learn more about how to improve our project:

* [www.geeksforgeeks.com](http://www.geeksforgeeks.com)
* [www.wikipedia.com](http://www.wikipedia.com)
* www.stackoverflow.com
* <https://cs.stanford.edu>