**Implementing a Search Engine that uses the TFIDFPageRank score to search a given query**

Uses a special score "TFIDFPageRank" that combines the functionalities of the TF-IDF measure and the PageRank algorithm's ranks to produce improved search results.

Name: K.Prahlad  
Class: B.Tech Computer Science 3rd Year, 2nd Semester  
Roll Number: 167227 (Section-B)

This project is an individual project as part of the course "Advanced Data Mining" at National Institute of Technology, Warangal as required for the Minor-2 Examination.

**Project Aims:**

1. To understand the basic working of a general search engine, and to implement TF-IDF and the Google PageRank Algorithm.
2. To search a given query on a given website, and rank all webpages in terms of a score that uses the TF-IDF of the query keywords with respect to the website corpus and the PageRank obtained from performing Web Structure Mining on the website.

**Project Inputs/Parameters:**

1. Maximum number of webpages to crawl in the website.
2. The total number of iterations to update PageRank.
3. The Damping Factor
4. The Query sentence/string

**Project Output:**

A sorted list of webpages, ranked from best to worst- to visit when a query is searched.

**Project Description:**

Most basic search engines have main 3 parts. Thus, my implementation has the following processes:

Crawler: A crawler crawls the web page, fetches the contents and links in the page and stores them. The developer of a website can define a file called Robot.txt that defines how frequently a crawler is allowed to crawl and which web pages it is allowed to crawl. The crawler program in this project works in a similar fashion, looking up new links exhaustively and storing their content. It also computes the graph (web structure mining) of the webpages, used for PageRank.

Indexer: Creates indices to store each word in the content of every URL. This process helps the calculation of TF-IDF and is also used to analyze and know the pages associated with the keywords in the search query entered.

Query Processor: This program takes the graph created by the crawler to calculate the PageRank of every document. To filter results according the user query and to calculate TF-IDF, the indices from the indexer are used. After the calculation of the PageRank of every document and the TF-IDF scores of every keyword in the search query with respect to each document, we then calculate the TFIDFPageRank score.

TFIDFPageRank of a document = (PageRank of the document) \* (sum of the TF-IDF scores of all search keywords present in this document + 1)

I designed the above formula based on reasons elaborated below:

1. The plain pagerank algorithm calculates the pagerank only based on the structural aspects of the website and the webpages. The pagerank of a page is independent of the search query.
2. Due to this, a clear disadvantage of not efficiently utilizing the search query for ranking is observed. The algorithm would just filter webpages that had the words in the query and sort them using their pageranks.
3. To add a contextual support to the pagerank, the TFIDF score would balance the above mentioned problem (when multiplied with pagerank), as the TFIDF score indicates the importance of a word in a corpus, and thus we would now also be concerned about whether the words in the query are important or not with respect to the webpages.
4. Thus, if important keywords in a search query are present in pages having a lower PageRank, they are given proper justice in the results, and vice versa.
5. The TFIDF sum is added with 1, because in some cases the TFIDF score may become 0 (when the keyword is present in all documents). If we directly multiply this value, the PageRank would be 0, which would underrepresent the rank of the page. Thus, adding one retains the pagerank whenever necessary.

After calculating the above score the Query Processor sorts and prints the results in descending order of TFIDFPageRank scores.

**Observations and Conclusion:**

It was observed that the algorithm works well for websites that have smaller number of webpages. This is because of the overhead of the TFIDF computation, that would need additional time and space.

Best number of webpages to crawl: <=40 (for my computer running an Intel i5 processor on 4 GB RAM)  
Best number of iterations: 1000 (PageRank values stabilized beyond this)  
Best Damping Factor observed: 0.85

As the damping factor was raised, the individual values of the TFIDFPageRank decreased, yet the ordering remained the same.

**Notes:**

By default, as an example the official Stanford NLP group website has been taken as the website in which we would search our queries. Link- [https://nlp.stanford.edu](https://nlp.stanford.edu/)

Please refer to the PNG files attached to this repository to check how the expected input and output should look like, with changing damping factor and number of iterations.

This project has been done individually.