Information Retrieval

Assignment 1

News Article Search Engine

Developers:

Ayush Laddha 2017A8PS0717H

Dhruv Gupta 2017A7PS0108H

Saarthak Jain 2017A7PS0083H

Smit Shah 2017A7PS0080H

**1.1 Objective:**

The aim of this Assignment is to develop a small version of a big Search Engine using the algorithms taught in class. This is a news article search engine where the user inputs a particular query and fetches the top 10 documents based on vector space model of information retrieval.

We are working on a dataset of almost 2200 news articles, comprising of 5 categories – Business, Tech, Sports, Politics and Entertainment. The objective is to find effectively the top 10 articles which adhere to the query entered by the user, and thereby returns the list of documents, along with the subcategory and the news headline for the article.

**1.2 Definitions and Acronyms**

The terms used in the document and their respective definitions are given as below:

• **Term Frequency** - Defines the number of times a word occurs in a given document.

• **Tokens** – Each word is taken as a token for processing the document.

• **Stemmed word** – Its defines the primitive form of a given word.

• **Stop word** – These words are usually the connectors and should not be considered in while processing a document. These will be removed in the pre-processing stage of the project.

• **Inverse Document Frequency** – Given a set of documents, idf of a word gives the number of documents in which the term occurs.

• **Document Frequency** -It basically tells the number of documents in which the word appears.

• **Collection Frequency**-It describes the number of times the word has appeared across all the corpus.

**1.3 System Architecture:**

The project is divided into several modules which take care of pre-processing, tokenizing, stemming, removal of stopwords, calculating the term frequency and inverse document frequency for all the terms in the documents.

We have not used Boolean model and explored the use of Vector space model, The representation of a set of documents as vectors in a common vector space is known as the vector space model and is a major part to a host of information retrieval operations ranging from scoring documents on a query , document classification and document clustering.

Using the single word queries gives results easily and in a way the output for single word queries are generalized and doesn’t present much important information retrieval across corpus.

In the pre-processing module, all the documents are taken to retrieve the tokens in each of them which are stored in an array and the words which are identified as stopwords by nltk stopwords are removed from the array as they carry negligible importance in this project.The remaining tokens are subject to stemming and removal of punctuations to get the primitive form of the token for effective computation of tf and idf.

When query is given as input the after applying all pre-processing steps on query we have made the vector of query Then from data present in structure of node we have calculated document vector for each relevant document and calculate the tf On each document, tf is calculated for each term and stored which is further used in calculating tf-idf score and while giving output we give output ranked by tf-idf score.

A screenshot of a cell phone

Description automatically generated

Data Structures used:

1) Dictionary

2) List

3) Set

4) List of Dictionaries

**Data pre-processing:**

* Dataset consists of approximately 2200 documents categorised into 5 folders. The test folder contains 3 of these categories each having 5 documents. This helps developers to test easily before running on the whole dataset stored in bbc folder.
* The dataset is read using the ‘os’ and ‘pandas’ library in Python. The ‘os’ library is used for fetching the current working directory path and iterating through each folder recursively. The ‘pandas’ library is used for storing the dataset in an ordered fashion. There are 5 major columns in the dataframe : headline, brief, article, type and filename.
* The dataframe is then read and for each row of the dataframe we preprocess the article. Preprocessing involves first lowercasing the article, using nltk library to tokenize the article, removing stop words and stemming the words to their root form using Porter Stemmer.
* The input query to be searched across the corpus is also similarly preprocessed. Along with this, a spell checker function is implemented which automatically corrects a wrong word in the query, if any.
* Next, once we have all the unique words present in the corpus ,we compute the tf-idf values for each document and also for the query.
* Tf score is computed simply by taking (1+log(term-frequency)).The Idf score is different for query and document.For a document ,the idf score is assumed to be 1.Whereas for a query,the idf score is computed using (1+log(Total number of documents/Document frequency of term)).
* The tf-idf score vector for the documents is stored in the form of a list of dictionaries,where each dictionary in the list corresponds to the tf-idf score of a document,and each dictionary consists of the words present in the document mapped against tf-idf score for the word in the document.
* Similarly,tf-idf score vector for the query is stored in a dictionary mapping query words to tf-idf scores.

**Computations involved**

* Once we have the tf-idf score vectors ready for all the documents and the user query, the next step is to find the top 10 documents which are closely related to the user input query.
* For this purpose,we will be computing the cosine similarities between score vector of the query and for every document in the dataset. Cosine Similarity is simply the normalised dot product value between the query and document vector.
* Once the cosine sums are computed, we sort the values in decreasing order and return the documents with top 10 cosine sums, mentioning the category,News headline and file name as output.
* Running Time for creating dataframe from the dataset-0.4479565000000001 seconds
* Running Time for storing all the unique words of the corpus- 29.496402399999997 seconds
* Running Time for computing tf-idf vectors of all documents and query- 28.3294051 seconds
* Running Time for computing cosine sums and returning top 10 results- 0.20988670000000553 seconds
* As is evident from the results obtained above, the major time wasted is in the process of finding the unique words in the corpus and computing tf-idf scores for all documents.
* Therefore if the user enters same query many times, all these computations will have to be done every time,in order to give the same result each and every time.
* To overcome this, we have implemented a **caching mechanism**, wherein the results of each query are stored in a text file.
* When the user enters a query,the first step is the check in this text file if the result for the same query exists or not.
* If it is found, the process ends there itself and prevents unnecessary computations.

**Results obtained for different queries**

1. Without Caching:

* ‘Decision’ – 61.3020797 seconds
* ‘coolest choreography event’ – 60.9728019 seconds
* ‘Mr Mittal told US investors that once the acquisition of International Steel Group’ – 66.8051068 seconds
* ‘The sequel to the best-selling Need for Speed: Underground has inched ahead of the competition to take the top slot in the official UK games charts. The racing game moved up one spot to first place, nudging GTA: San Andreas down to second place. Halo 2 dropped one place to five, while Half-Life 2 fell to number nine. Last week's new releases’ – 67.0945468 seconds

1. With Caching:

* ‘Decision’- 0.19907720000000007 seconds
* ‘coolest choreography event’ – 0.1850121 seconds
* ‘Mr Mittal told US investors that once the acquisition of International Steel Group’- 0.1884516999999999 seconds
* ‘The sequel to the best-selling Need for Speed: Underground has inched ahead of the competition to take the top slot in the official UK games charts. The racing game moved up one spot to first place, nudging GTA: San Andreas down to second place. Halo 2 dropped one place to five, while Half-Life 2 fell to number nine. Last week's new releases’ – 0.2640334999999998 seconds