DOMAIN SPECIFIC INFORMATION RETRIEVAL SYSTEM

OPERETTA - A LYRICS BASED SONG RECOMMENDER SYSTEM

OBJECTIVE

The project aims at making **a lyrics based song recommender system**, where the **top ten most relevant** songs will be retrieved based on the given query. The dataset is taken from **musixmatch**

It is a highly varied and huge dataset containing more than two hundred thousand songs in various languages with majority being in English.

DATASET ARCHITECTURE

The dataset is already decomposed in bag of words model. We are given two files. The dataset decomposition is as follows:-

1. File 1 : The **top 5000 words** which occur in 92% of the dataset are given.
2. File 2 : The list of all songs is given, and each song is given in this format
   : **song id**
   : **frequency of each of the 5000 words** corresponding to the song

PRE-PROCESSING

The technique used for pre-processing is forming an **inverted index** corresponding to the **bag of word** models provided via the dataset. Three new files are created to store the necessary information after completing the pre-processing in order to avoid the overhead of repeated pre-processing each time the script is called.

1. dict.txt : A dictionary of indices corresponding to the 5000 words where corresponding to each word we have a dictionary which gives the frequency of appearance of that word in each song. Basically this is the inverted index of the dataset.
2. names.txt : A list of names of the songs for the corresponding index of list.
3. Freq.txt : A list which contains the number of documents a word has appeared into, corresponding to each word.

INVERTED INDEX (dict.txt format explained) :

f^{IJ} : frequency of word id I in doc id J

{ {word id 1}->{{docid1 -> f^{11} }, {docid2 -> f^{12}}, ..}, {word id 2}->{{docid1-> f^{21}},{docid3 -> f^{23} } ..}, ..}

MODEL

For each incoming query **term frequency** and **inverted document frequency** are calculated, which are then compared with the query. **Cosine similarity** is used to find the top ten most relevant results with respect to the query.

CODE FLOW

1. User enters a query.
2. **Stemming** and **lemmatization** is performed on the query. This code is the exactly similar to the one used by the dataset creators to create the bag of word models to avoid any sort of discrepancy between the query and dataset. Function corresponding to this facility is **query_processing().** The function return a list of query words.
3. The **top_ten_given_query()** function takes this list of words and calculates the top ten songs pertaining to the query.
4. TF-IDF is calculated for each query word corresponding to each document. This is an unavoidable overhead since the query word are required for generation of TF-IDF. Document vectors are constructed using this, that is vector for each document.
5. Cosine similarity is calculated between the document and query vectors.
6. The top ten results are printed.

TIME AND SPACE COMPLEXITY

W : Total Words in dataset

D : Total Documents in the dataset

A : Average Number of Words in a document

Q : Total Words in the query

PRE-PROCESSING TIME :: O(W * D * A)

QUERY-PROCESSING TIME :: O(Q* D)

QUERY-PROCESSING SPACE :: O(Q*D)

AUTHORS

ABHISHEK BAPNA : 2018A7PS0184H

ASHNA SWAIKA : 2018A7PS0027H

HARDIK PARNAMI : 2018A7PS0062H