DOCUMENTATION

Team:

Sumanasa Somu	2017A7PS0114H
J.S.N.S Rahul	2017A7PS0262H

Aim:

To make a search engine that retrieves relevant documents when a query is entered using the vector space model and tf-idf along with Positional Indexing score for ranking.

Dataset Used:

The dataset for the search engine is obtained from the Kaggle. It consists of "Song name", "Artist", "year of production" and the "lyrics". The dataset is of the format CSV(Column separated Vectors) with 5 columns - Doc-Id, Song, Artist, Year, Lyrics.

Modules/Frameworks Used:

- nltk - Natural language tool-kit used for tokenizing and stemming
- math - To perform mathematical operations like logarithmic operations and finding the square root.
- csv - To handle the CSV File (Dataset)
- flask - A web framework used to build the application.

Functions:

readDataSet(): Takes data from the dataset and appropriately creates lists/maps/dictionaries of original words and also performs stemming operations for normalization.

update_tf_idfTable(doc_stemmedTerms_Table, terms): Produces the tf-idf table using the terms and documents.

positionIndex(docId, data): *Updates the positional index of given document for all the terms along with frequencies*

editDistance(str1, str2, m, n): *Returns the least no. of operations to convert a string str1 of length m into another string str2 of length n using dynamic programming.*

getNearestTerm(terms, word): *Returns a term from all terms requiring minimum number of operations to convert to the given word.*

getWordWithSuffix(terms, word): *Returns a term from the terms which has the given word as its suffix.*

proximity_score(input): *Returns the score calculated based on the proximity index*

total_score(query_terms): *calculates the total score giving 70% weightage to tf-Idf and 30% weightage to positional score*

getResults(query): *Returns the most relevant search results for the given query.*

getDotProduct(a, b): *Returns the sum of the product of the corresponding elements in the given lists a, b.*

index(): *Returns the main home page with the search bar.*

fetchDocumentDetails(doc_id): *Returns the document with the given document ID(doc_id).*

searchResults(query): *Returns the page with top ten relevant search results.*

displayDoc(docname): *Returns the document with the same name/Id as "docname".*

GUI: (Front-End)

layout.html: Basic layout of the web pages (includes the stylesheet) and is an extension to other pages.

home.html: Displays the home page containing the search bar and shows the search results upon a search.

searchResults.html: Displays the search results in the form of a list containing document name/ID.

document.html: To display the document which the user wishes to see by clicking.

_navbar.html: To display Navigation bar

_messages.html: Used in displaying the error/flash messages