Design Document

# Data Set:

The dataset consists of about 5000 documents. Each document consists of ‘DocID’,

‘Song name’, ‘Artist’, ‘Year’ and ‘Lyrics’. This dataset is in CSV (comma separated values)

format. The dataset was obtained from Kaggle.

# Technologies used:

* Python
* BootStrap for Styling (CSS+JS)
* Flask Web framework
* Other Python libraries which include nltk, wtform, pandas

# Implementation:

**Model Used** : Vector Space Model and tf-idf weights for ranking

The steps of constructing data structures and producing ranks of the documents in the search

result are as follows.

The dataset is read using python - CSV library. The text in each document is then tokenized and each term is stemmed using python - nltk porterStemmer. A dictionary is constructed while reading the dataset with keys as document names/id and values as the list of processed i.estemmed terms of its text.

**doc\_stemmedTerms\_Table** = {DocName : [ stemmed\_term1, stemmed\_term2, ..….]}

Also, all the processed terms are inserted into a list during this step. The list is then converted

into a set to remove duplicates. For every term in the set, the documents in which contain the

term and the count in the document is stored as a dictionary. This dictionary is added to invertedindex (another dictionary) with the key as the term.

**invertedIndexTable[i]** = {stemmed\_term : { doc\_id : count}}

The tf-idf matrix (no.of docs \* no.of unique-terms) is constructed. The formula for computing

tf-idf value of a term in a document is given by (1+log(tf)\* log(N/df)) \* (cosine-normalisation)

tf is the frequency of the term in the document. df is the number of documents containing the termand cosine-normalisation is equal to sqrt(sum(square((1+log(tf)\* log(N/df)))))

**pos\_Index =** {docid : { term : count,{poslist}}}

The positions and frequency of the terms in the document are stored to calculate the positional score later for retrieving the relevant documents using total score

**tf\_idf\_Table =** [ [ (tf-idf)…..].....]

A query is taken as a document, tokenized and stemmed and tf-idf values are calculated and

stored as a list. If any of the words in the query are absent in any of the documents, its nearest

word substitutes in the query.

[(tf-idf).....] for the query

By multiplying and adding the corresponding tf-idf values with each document, the search query is scored. Top 10 relevant documents are returned. This is done by sorting the scores in inverted order. for query vector ( (1+log(tf))\* log(N/df)) where N increases by 1 for the query

**SearchResult =** [doc\_1, doc\_2, doc\_3……, doc\_10]

The relevant 10 documents are shown and if the word is not found/ documents are not relevant

then the rest are filled according to the doc\_id.

# Data Structures:

**Dictionary** : Inverted Index Table is constructed as a dictionary of dictionaries.

stemmed\_Term = { stemmed\_term: { docID : termCountInTheDocument}}

*Advantage* : Uniqueness and key value pairs

**List 2D** : tf-idf weights are stored as a Matrix

*Advantage* : Easy to fetch using indices.

**Set** : Unique stemmed terms from the corpus for suffix search and spelling mistakes or

closer-word(require less changes to be made) searches.A*dvantage* : Uniqueness.

Time:

10 documents

Reading Documents: 0.0575s

InvertedIndex construction 0.0077s

Table constructions 0.0008s

retrieval of relevant documents 0.00825s

100 documents

Reading Documents: 0.4611s

InvertedIndex construction 0.2941s

Table constructions 0.0336s

retrieval of relevant documents 0.08056s

250 documents

Reading Documents: 1.121s

InvertedIndex construction 1.352s

Table constructions 0.152s

retrieval of relevant documents 0.2543s

500 documents

Reading Documents: 2.387s

InvertedIndex construction 5.335s

Table constructions 0.537s

retrieval of relevant documents 0.7142s

1000 documents

Reading Documents: 4.876s

InvertedIndex construction 18.69s

Table constructions 2.008s

retrieval of relevant documents 2.2954s

# Group Members:

Nitin Chandra Surapur -2018A7PS0188H

Mohit Guddanti -2018AAPS0321H

Abhinava Maddha -2018A8PS0844H