<u>Domain Specific Information Retrieval System</u>

Objective :-

The project is to retrieve top ten matching songs based on the query given by the user from the database containing more than 2,10,500 songs and also the top ten most popular songs till date.

Architecture:-

The database consists of two parts.
1)List of top 5,000 words that occur in all the songs which account upto 92% of all the unique words
2)Track IDS of all the 2,10,500 songs along with the words and their corresponding frequencies occurring in that particular song are given

Inverted index was created so as to rearrange the data according to our needs(Boolean matrix was not possible due to size constraints)

Primary structure of Inverted index:

WordId -> { DocID: count -> DocID:count -> DocID : count … }
Where
DocID represents the IDs of the documents(songs) in which that particular word is present
Count represents the frequency of the word in that particular DocID

The inverted index was stored in the form of dictionary of dictionaries as explained above

Lists and dictionaries were used at multiple occasions as required in the code

Eg : List was used to store the Track IDs for each song e.t.c

Preprocessing:

1) Creating the inverted index
2) Total words in a song

Flow of code :-

- User enters the query
- The query is tokenized into words
- Words not present in the corpus are ignored
- Tf-idf is calculated for each document for each query word(calculation of tf can’t be preprocessed as inverted index is used instead of boolean matrix due to size constraints)
- vectors are prepared for each document based on tf-idf values along with the query vector.
- Cosine similarity is used to compare each document vector with the query vector
- Top ten document vectors most similar to the query are retrieved and their corresponding Track IDs are retrieved
- The results of a query are stored in a frequency list which will in turn facilitate retrieval of the top ten most popular songs till date

Running Time:

Preprocessing Space Complexity :- $O(w*i)$
Query Space Complexity :- $O(n*q)$
Time complexity :- $O(q*i)$
Where
n= Number of documents(songs)
i=Average number of documents in which a particular word occurs
q= Number of words in that query

w= Number of words in the corpus