**SONG LYRICS SEARCH ENGINE**

**The data set:**

We used the Musixmatch million songs dataset to train our retrieval model. It consisted of formatted data of 27143 different songs with their unique track IDs, term frequencies of the overall top 5000 terms, that constituted roughly 92% of all occurrences in the entire corpus of lyrics available. The dataset was taken as a TXT file.

**Technologies used:**

Programming Language- Python 3.6

Libraries- Porter2stemmer (for normalization), Pandas(for reading files), NumPy (for working with arrays), CSV (for creating file that stores tf-idf scores), Math (for logarithmic and other calculations).

**Implementation:**

The search engine is essentially a ranked retrieval system in which vector space modelling technique has been implemented. The steps used in building the search engine are as below:

1. The dataset containing track id’s of all 27143 songs was read into a python dictionary with keys as the track ID’s and values as lists containing terms and corresponding raw term frequencies as available from the dataset. There wasn’t any need of normalization here as the dataset itself was prepared with terms normalized using a technique similar to porter stemming.
2. The IDF scores of all the songs with respect to all the terms were computed as per the formula IDF= log(N/DF) if DF is nonzero and a very large value otherwise, where N is total number of documents(songs) and DF is the raw document frequency of the corresponding term. Owing to the large size of our dataset, all the IDF scores were stored in a CSV file.
3. The TF scores for all the documents (using the formula TF=1+log(raw term frequency) if term is present and zero otherwise) and hence the TF\*IDF scores were computed and stored in another CSV file.
4. Once a query is entered, it is tokenized and normalized using the same technique that has been used to create the dataset. Now it is treated the same way as the other documents(songs),i.e., its TF-IDF vector is created in the same fashion.
5. In order to find similarity between the query and the documents, an improved square root cosine(ISC) similarity is used as proposed in the research paper by Sahar Sohangir and Dingding Wang, titled [Improved sqrt-cosine similarity measurement](https://link.springer.com/article/10.1186/s40537-017-0083-6). The similarity scores are sorted in descending order and top 10 most similar items are returned.

**Data Structures used:**

Dictionary: In order to store the song info from data set in the form track ID: string containing list of term frequencies

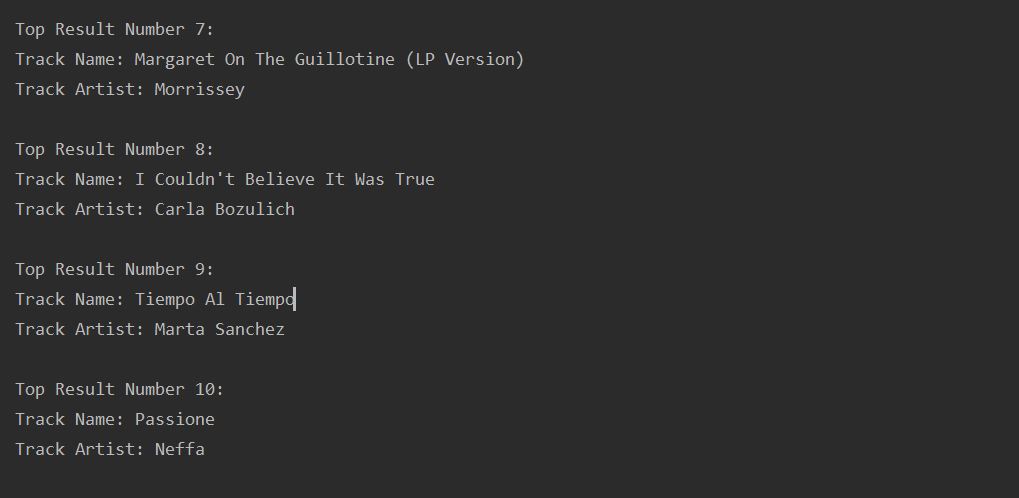
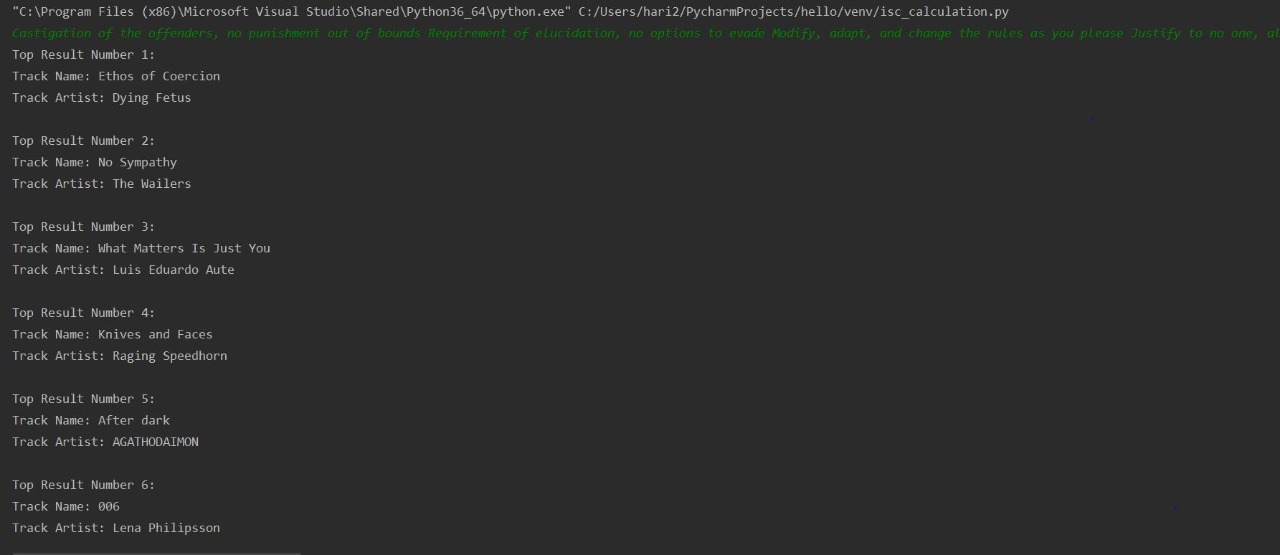
List: For temporarily storing IDF values in order to calculate TF-IDF scores, to store TF-IDF scores before writing them into a file, to store all components of query vector and also to store similarity scores (which would be sorted and top 10 values would be returned).

**Running time:**

**#Sample run no 1.**

**Lyrics of song ‘Ethos of Coercion’**

**Run Time: 90 sec**



**#Sample run no 2**

**Lyrics of song ‘No Sympathy’**

**Run time:105 sec**

