Music Information Retrieval
An analysis of designing a music retrieval system

Serfin Hernandez
Cesar D. Quihuis-Romero
Ping Tontrasathien

CSC483 - Information Retrieval
4/24/2025

Introduction

Information retrieval as the name suggests is the process of retrieving information pertaining to a given query from a repository of information. In the case of information retrieval for musical queries the challenges broaden in scope depending on the form of the given query. That is musical queries can come in the form of both sound and written queries. For our system, we will not be handling sound comparisons or analysis, as waveform analysis requires much more laborious work to clean up noise and to create a clean mapping. For our purposes we will handle simple text based queries.

For this purpose developing a simple information retrieval system which could handle a text query, the problem is still complex and comes with similar challenges found in book, article, or source based information retrieval. Many of these challenges come from the form of the text based query, since there is no way, other than administrative control measures to mitigate users placing any number of tokens in any number of assorted combinations. Additionally, many of the tokens are optional and may be omitted. These tokens include: artist name, year of publication, album name, featured artists, and lyric(s).

A robust system would most likely rely on a database, where the user can use filtering tools to provide additional information regarding their query, and their keyword search would only be relevant to the desired lyrics. Doing so would mitigate and outright eliminate the problematic parsing of complex strings, which contain numerous tokens which given context could be interpreted in a multitude of ways. This especially, needs to be considered when faced with lyrics themselves which contain dates, names, and references to other elements which would mistakenly be parsed as filterable items.

Using an interface would allow for a separation between items that function as filters while not influencing the query which would only contain song lyrics. Utilizing filtering would also positively impact performance, allowing for more optimized search queries that have a smaller subset of the whole database to compare against rather than a database containing millions of songs. Differentiating filtering items versus song lyrics is not the only unique challenge facing music information retrieval.

Additional challenges are faced when parsing, tokenizing, lemmatization, and normalizing the corpus of song lyrics. As much of spoken and sung English differs from proper and written english. That is, many artists and genres of music rely on forced rhyming, broken or improper English, as well as, unique or self made terms. This means that when tokenizing special attention should be placed to optimize parsing to not incorrectly omit song lyrics which at face value may not look like an accurate term or token.

A naive and more simplified approach to separating song lyrics and filterable elements would be to implement simple administrative controls, by way of partitioned user prompts. One representing lyrics, one each for subsequent requested filters (i.e. year, album, artist, etc). Well simplified, this approach proves to be more “bulky” and “clunky” and places the burden on the user to understand the prompts and interact with them proficiently. This unintuitive prompt interaction and reliance on user understanding, leads us to push to implement an easier to use interface.

Doing so as discussed previously, would provide a clear delineation of user intent, easier parsing of lyrics, as well as, year, and artist filter. Implantation of such an interface does come with extra work and more information to validate and either utilize or dismiss if users omit. This front end cost of developing additional features and having strict field checking and validation, is offset in both ease of use and a higher likelihood of proper user interfacing, accomplishing the goal of giving the user access to relevant information.

Methods

Our goal was to perform a comprehensive analysis of the file that contains the songs' metadata in order to retrieve the most relevant information which in our case is the most relevant song based on the user's query. The query consists of the following partial lyrics, year, and singer.

To achieve this analysis we used the data in the form of a .csv approximately around 3 million songs

- song_lyrics.csv

From this .csv file, we generated a data frame with the following features:

- Title (Name of the song)
- Tag (Genre of the song)
- Artist
- Year
- Lyrics
- Id (Unique id of song)

In order to generate the data frame we first dropped columns such as

- language_cld3
- language_ft
- language
- features
- views

After dropping the columns we explored if there were any missing values which would pose a problem when indexing into the dataframe. Then after dropping columns that would be NA we then explored the lyrics to be able to parse through the lyrics. Upon exploration we found the the lyrics contained phrases such as "[chorus]" and "[verse #]" where the "#" denotes the numbered verse/chorus. The lyrics also contained punctuation that would interfere with the parsing of the lyrics and would yield non-lookable words when we query the file. In order to resolve this issue we used the NLTK (Natural Language Toolkit) Python library in order to parse the lyrics and lemmatize the lyrics. After the transformation the lyrics were in the python list in the dataframe.

In order to retrieve the songs first we intend to filter the songs as the full corpus is expected to be around 3 million songs and running the tf-idf algorithm on the entire file would make the program slow. Therefore in order to circumvent the need to run the algorithm over the entirety of the file the data frame will go through two filters. The filters would be determined by the query; the first one would be year and then second would be the artist's name. If the filters are not predefined then the algorithm would be run on all songs. For faulty filter parameters such as a year that does not exist in the data frame or an artist that does not exist the program is to treat the parameter as if it were not defined and continues to the next filter. If both filters are bad then the program continues on the entire dataframe. The algorithm is expected to return the top 10 songs that are most relevant to the query. Sorted by cosine similarity and returned as a dataframe using the id's.

For the next part of the program we used the Openai Python library the model being used is "gpt-4o-mini", this model was used as the mini model is more compact and uses less computing power than the full "gpt-4o" model. For the first iteration we asked the model to rerank the top 10 songs for ease of the model. The data frame with the top 10 songs returned by the tf-idf algorithm is turned into a .json file as the openai model is not able to take a data frame as an input. The prompt used for the reranking is as follows

- Instructions

  "You are a reranking tool that takes in some already ranked songs in a JSON file. When solving the query, give the result as song - artist (one per line)."

- Input

  f"Rerank the following songs using this query: {query}\n\n{top10_json}"

For the input the {query} is the initial query string that the user inputted and {top10_json} is the top 10 songs from the tf-idf algorithm that were turned into a JSON file. For the second iteration we ingested the top hit song in order to get a small description of the song for the user. The prompt used to yield a description of the top hit is as follows

- Instructions

  “Your objective is to give me song info (artist, year, album, brief history) in less than 100 characters the song name is {song name} the artist is {artist name} and the year is {year}”

- Input

  “Tell me about the song titled {song_name} from {artist_name} that came out in {year}”

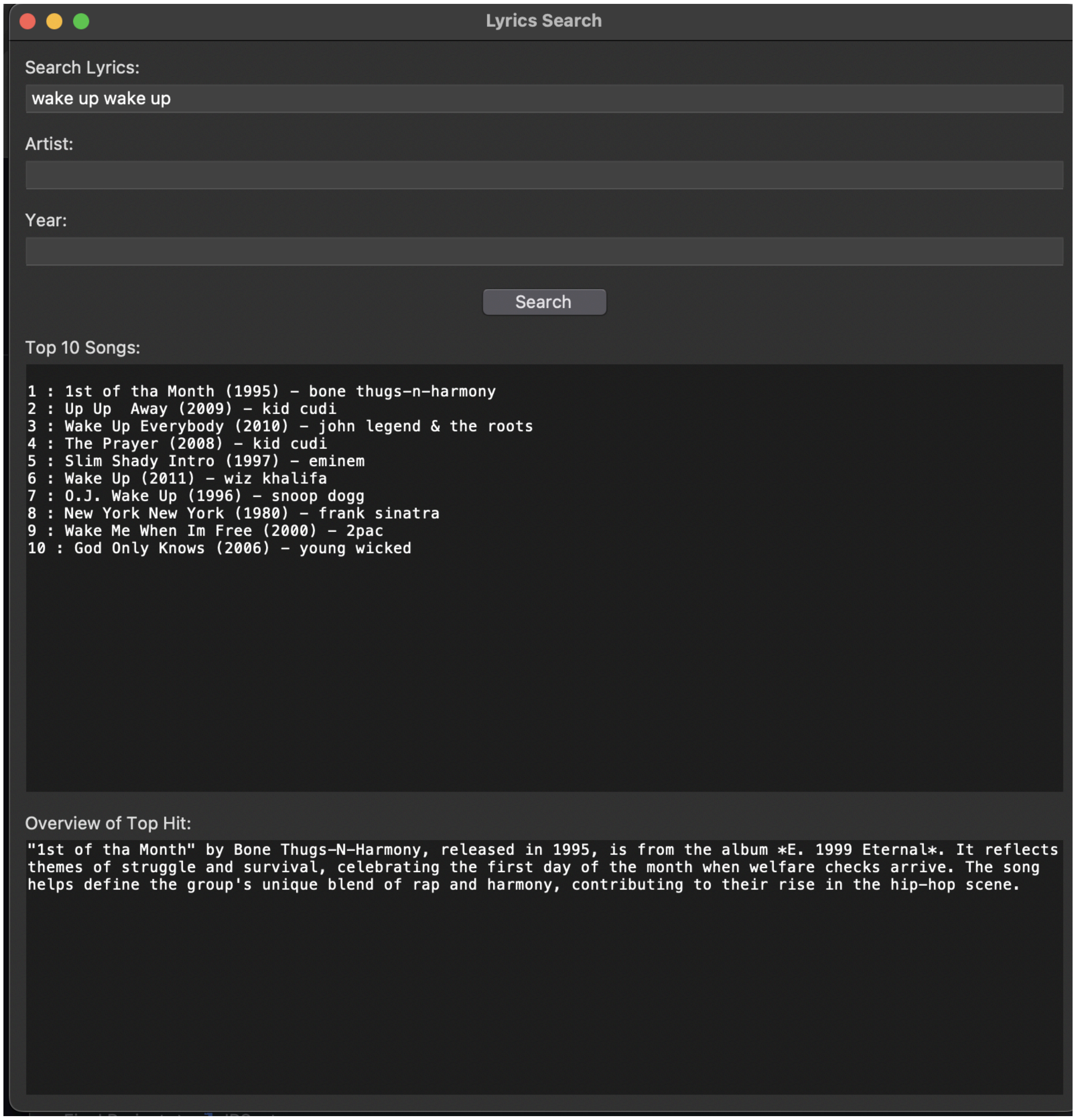
{song_name}, {artist name}, {year} are from the first row in the top 10 dataframe title,artist,year respectively. The model uses its database to give the description based on our parameters

Result

The result from querying will be printed as a numbered list of 10 songs. The first song on the list is the most relevant song. The higher number means the lower relevance. It is possible that this IR system returns irrelevant songs, if the searching query does not appear anywhere in the database or the number of relevant songs is less than 10.

The format of the output is: Rank : Title - Artist.

For example,



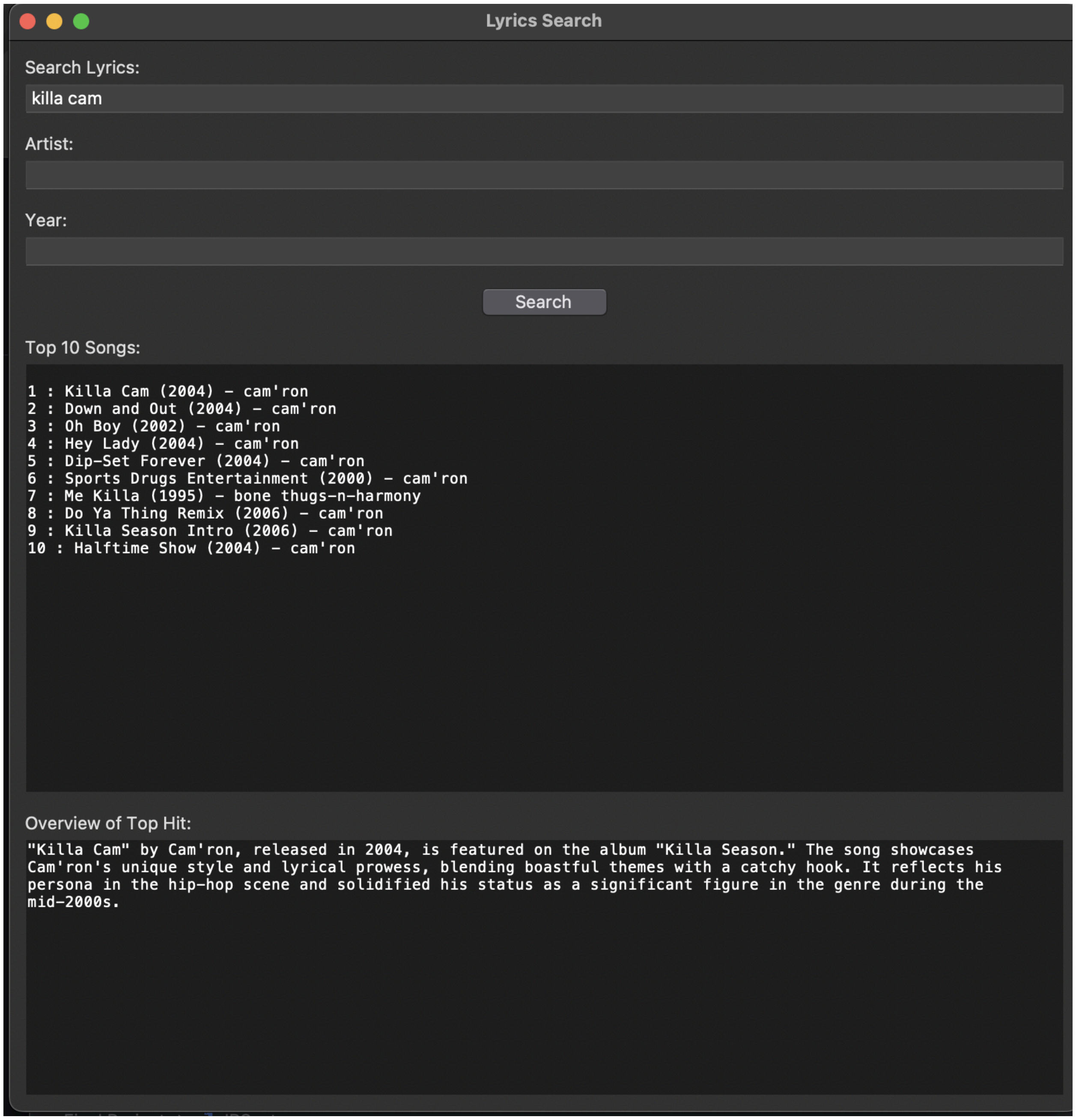
Search Lyrics: wake up wake up

```
1 : 1st of tha Month (1995) - bone thugs-n-harmony
2 : Up Up  Away (2009) - kid cudi
3 : Wake Up Everybody (2010) - john legend & the roots
4 : The Prayer (2008) - kid cudi
5 : Slim Shady Intro (1997) - eminem
6 : Wake Up (2011) - wiz khalifa
7 : O.J. Wake Up (1996) - snoop dogg
8 : New York New York (1980) - frank sinatra
9 : Wake Me When Im Free (2000) - 2pac
10 : God Only Knows (2006) - young wicked
```

```
"1st of tha Month" by Bone Thugs-N-Harmony, released in 1995, is from the album *E. 1999 Eternal*. It reflects
themes of struggle and survival, celebrating the first day of the month when welfare checks arrive. The song
helps define the group's unique blend of rap and harmony, contributing to their rise in the hip-hop scene.
```

However, if users specify year and artist, the database would be filtered and could be left with less than 10 songs. The list would not have 10 items, but have as much as how many songs that are left within the filtered database. When the user is using the filtering for the artist option it is case insensitive but they must be aware if the artist has any punctuation such as apostrophes or hyphens or else the filter will not work since we are not lemmatizing the artists' name.

Once we have a list of retrieved songs, the first song on the list or what we call "top hit" will be detailed by "gpt-4o-mini". It will tell brief information about the song, such as, the year the song was released, album name, and some history of the song for example when the song is "Killa Cam".



```
1 : Killa Cam (2004) - cam'ron
2 : Down and Out (2004) - cam'ron
3 : Oh Boy (2002) - cam'ron
4 : Hey Lady (2004) - cam'ron
5 : Dip-Set Forever (2004) - cam'ron
6 : Sports Drugs Entertainment (2000) - cam'ron
7 : Me Killa (1995) - bone thugs-n-harmony
8 : Do Ya Thing Remix (2006) - cam'ron
9 : Killa Season Intro (2006) - cam'ron
10 : Halftime Show (2004) - cam'ron
```

```
"Killa Cam" by Cam'ron, released in 2004, is featured on the album "Killa Season." The song showcases
Cam'ron's unique style and lyrical prowess, blending boastful themes with a catchy hook. It reflects his
persona in the hip-hop scene and solidified his status as a significant figure in the genre during the
mid-2000s.
```

Furthermore there are still some optimizations that could be made to the model. An integral part of the runtime is held by the reading of the csv. In the original dataset we had 3 million songs to read the csv. It took approximately 10 minutes of runtime. In order to have a user friendly implementation we truncated the file to the first 999,950 lines which yielded us 12297 songs. For this truncated csv the read runtime was 2 minutes and 20 seconds. This is sub optimal but some alternatives are turning the csv into a parquet file and generating the dataframe using the parquet file or using the pyarrow engine for the .csv file. Also some shortcomings of the model our first iteration of the ai implementation when asking it to rerank it would copy and paste the original reranking therefore, we decided to pivot into the second iteration which had better performance in order to give an actual description of the song we had to feed the model the title,artist,year or else the model would output incorrect information upon verification. Since the ai model creates a new query everytime we run the algorithm some song descriptions may vary.

In conclusion the model underwent through two iterations in order to have the final version. We went from having a terminal program that added administrative constraints in order for the queries to be passed correctly to a GUI being implemented in order to control the user input and not burden them. Although the information retrieval system is accurate most of the time it will return random songs if the lyrics are not specified or if the lyrics do not exist in the corpus. As well when inputting a year wrong or an artist it must be valid or else it will only be based on the lyrics. Additionally, these problems are exacerbated when a positive filtering occurs (i.e. a valid artist or year is provided), but lyrics do not match. That is it returns the number of songs that the artist has in the corpus, which may be less than 10, but never more.