Songs Information and Lyrics

Afonso Caiado, Diogo Martins, José Maçães

up201806789@up.pt, up201806280@up.pt, up201806622@up.pt

January 20, 2022

Abstract

Nowadays, with the increasing use of streaming services, listening to music is just a click away. With the huge amount of variety that is available, what we choose to hear can say a lot about us, our own personality and our feelings.

Thus, in addition to the presentation of the lyrics, genre and album to which a song by one of these singers or bands belongs to, it will be possible to explore the same data in order to find specific words in them.

Therefore, this report focuses on the development process of our information system, which main goal is to allow users to perform various queries to the dataset, describing step by step each stage such as the treatment of the data, its characterization, the domain conceptual model, and after we started using *Solr*, the collection composition, query execution, evaluation and the improvements we made in the last phase of our project.

Keywords: Music, Songs, Lyrics, Data Processing, Pipelines, Solr, Schema, Query Execution, Evaluation.

1 Introduction

With this report, our goal is to provide information as detailed as possible of our information system that presents data about songs such as its name, artist, genre, lyrics and more. The chosen theme is interesting to us and feels relevant since most people listen to music and a search system regarding that could appeal to them. In the first part of the project, we focus on details about the dataset we prepared, like its its characterization and extraction, the correspondent data processing pipeline and data visualization. In the second part of the project, we approach information retrieval, the indexing process and the evaluation results, working with queries to allow the search process. All of the sections contemplated are essential to come up with our final objective that is an information system related to songs by the most popular artists of the last decade. At the last phase of our project, we improved some aspects that we felt that could have some improvement. With this project, we expect that we ca get to know more about search systems and information retrieval.

2 Datasets and Data Cleaning

The data we are using comes in .csv format, which is easy to manipulate with "Pandas" and "Python". We found lots of datasets in Google Dataset Search, but only the following presented the type of data that we wanted since the others had lots of songs by artists we are not interested in and very few of the ones we wanted, or little information about songs (not including lyrics, date and genre).

2.1 Song Lyrics

1. *Origin:* The Song Lyrics dataset is a dataset by Deep Shah, being the source of it Genius, a digital media company and website that allows users to provide annotations and interpretation to song lyrics. This dataset was fetched from Kaggle.

2. *Description:* This dataset is composed of 21 .csv files, each one about a different top artist from the last decade. These files range from 100 entries to about 1000 entries, having 6 columns each, which are "Artist", "Title", "Album", "Date", "Lyrics" and "Genre".

3. *Treatment:* We decided to keep all 21 of the .csv files and combine them, as the data in each one is organized the exact same way, the major difference being that the songs in a file are all from the same popular artist. We removed one column from the files when we combined them, the one related to the year of release of the song, since it was redundant because we already have that information in the column "date" in the format "YYYY-MM-DD" that refers to year, month and day. We consider that this dataset was a good choice since it gathers all the songs by the most popular artists in the last decade, so the use of the information system we are developing would be more relevant to the general population that listens to music.

2.2 Spotify Past Decades Songs Attributes and Spotify - All Time Top 2000s Mega Dataset

1. *Origin:* The biggest audio streaming and media services provider right now, Spotify, has an open

API, with the type of information that we desire available. Thus, Spotify Past Decades Songs Attributes is a dataset by Nicolas Carbone, and Spotify - All Time Top 2000s Mega Dataset is a dataset by Sumat Singh, both finding its source in Playlist Machinery, which is a website that allows users to organize songs on Spotify by many parameters, and, in the case of both datasets, the parameter used to gather the songs was popularity, so, in an indirect way, its source is Spotify. Both were also fetched from Kaggle.

2. *Description:* The first dataset was composed of 7 files with 15 columns each. Each file corresponds to a decade of music. The second dataset is composed of only 1 file with 15 columns about the 2000 most popular songs from the 2000s and 2010s. These files contain data about the most popular songs of the respective data, including genre, which was missing in the first one.

3. *Treatment:* First, we decided to keep only 2 out of the 7 files from the first dataset of this section: those from the last 2 decades. This decision was made because our main dataset (from the previous subsection) is composed by the most popular artitists from the last decade, so there was no use in having 5 more files containing songs from artists that weren't going to match with those on our main dataset since they only released songs in the 2000s and 2010s. We also decided to reduce all the files from the 2 datasets in this subsection to the 3 columns we needed: Title, Artist and Genre. These datasets were fetched purely with the intent of knowing the Genre of the songs in the Song Lyrics dataset.

3 Data Merging

After the selection and refining of the datasets, we ended up with two major datasets. They were obtained with the *Pandas* tool and *Python* scripts.

3.1 Combined Lyrics

This is our main dataset and it contains 5 columns after refinement:

- *Artist:* name of the singer or band to whom the authorship of this song belongs to
- *Title:* title of the song
- *Album:* name of the album to which the song belongs to
- *Date:* date when the song was released
- *Lyric:* lyrics of the song

3.2 Song Genres

This dataset informs us on the genre of songs and has the following information:

- *Artist:* name of the singer or band to whom the authorship of this song belongs to
- *Title:* title of the song
- *Genre:* genre of the song

4 Pipeline

To process the described data, the following pipeline was used:

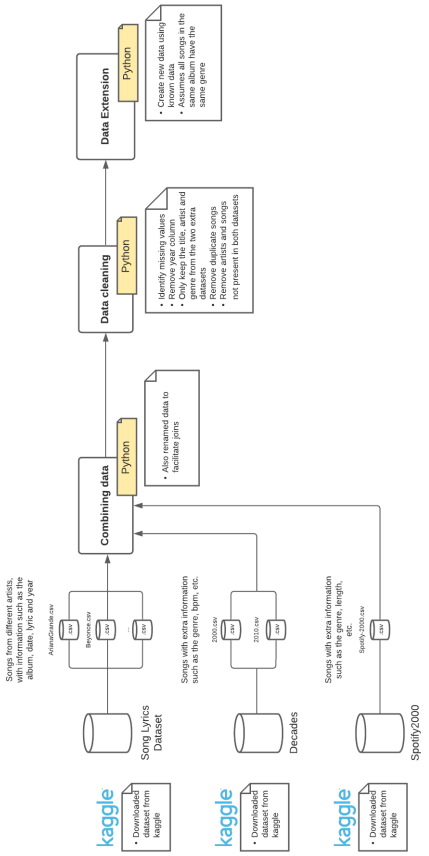

Figure 1: Data Processing Pipeline

We start by downloading 3 datasets, both from kaggle: the Song Lyrics Datatset, Spotify Past Decades Songs Attributes (named Decades in our diagram) and the Spotify - All Time Top2000s Mega Dataset (named Spotify2000).

With Python scripts, we combine the data merging of files and renaming the data to facilitate joins. Our next step corresponds to Data Cleaning, where the data was refined by identifying missing values, removing the "Year" column that came from the first dataset due to reasons explained in the subsection "2.1.3". We only kept the title, artist and genre from the other two datasets, so columns like bpm, loudness and energy were removed since they don't have a purpose for now in our project. Duplicate songs and the songs that were not common to our main dataset were also removed.

Finally, we proceeded to Data Extension, also using Python scripts, by creating new data using known data, assuming all songs in the same album have the same genre.

5 Domain Conceptual Model

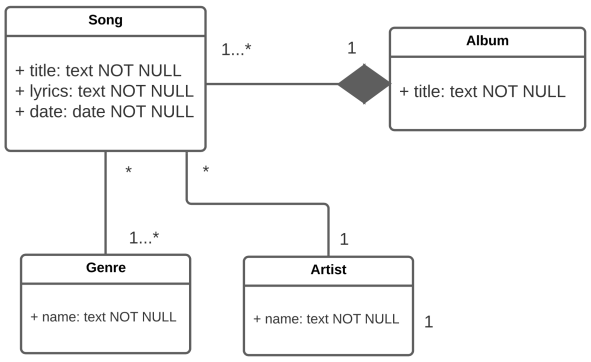

Figure 2: Domain Conceptual Diagram

The domain conceptual diagram consists of very few classes since we are only focusing on some attributes of a song for now. Our main class is the *Song* and it has the following attributes:

- "title" is the name of a song. Each song has a title but it is not necessarily unique;
- "lyrics" corresponds to the lyrics of a song;
- "date" corresponds to the date the song was released and it is not unique.

The other classes help complement our main class with additional information. This other classes are: *Artist*, which has only one attribute:

- "name" corresponds to the name of the artist.

Our dataset Song Lyrics is organized by artist and each song has a single artist associated with it, the "main artist" in case the song is a collaboration.

Album , which has only one attribute:

- "title" corresponds to the name of the album.

Since a song corresponds to a single album (the one it was originally released on) we represented that the song could only be present in a single album and not more than that one. Finally, we have the class *Genre*, which has also only one attribute:

- "name" corresponds to the name of the genre.

We assumed that a song couldn't have more than one genre.

6 Data Characterization

To better summarize some features of our data that we consider relevant for our project, we proceeded to provide the images explained below. We can see the most common words in lyrics of songs by Ed Sheeran, Beyoncé and Justin Bieber respectively (these specific graphs were provided by Deep Shah, Kaggle). For example, in Figure 5, the most common words in Ed Sheeran songs' lyrics are "time", "like", "go", "love" and "never".

Figure 3: Most common words in Justin Bieber songs lyrics.

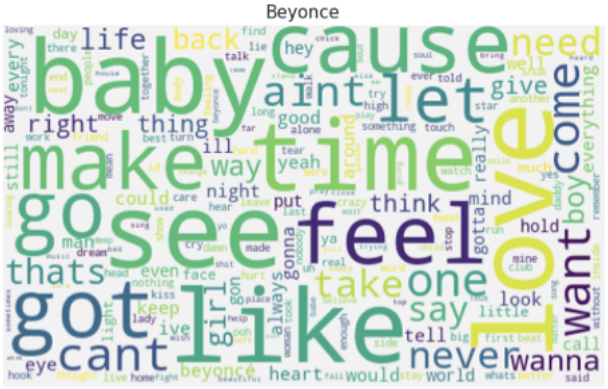

Figure 4: Most common words in Beyoncé songs lyrics.

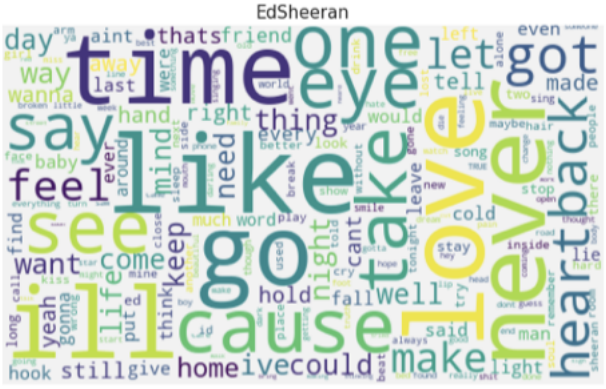

Figure 5: Most common words in Ed Sheeran songs lyrics.

In the next image, we can compare the amount of songs available in the dataset per artist in a graph. Eminem, Drake and Taylor Swift have more songs available and Khalid, Cardi B and Charlie Puth are the ones that have the least, which makes sense since they started their career way earlier than the others.

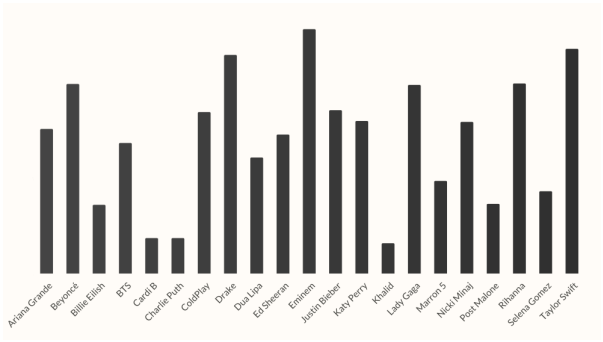

Figure 6: Amount of songs per artist in the dataset.

7 Possible Queries

Initially, in order to extract information from the data, we thought of some queries that would be interesting to present:

- Search songs by artist
- Search songs by date
- Search songs by genre
- Search songs by album
- Search songs by name
- Search songs by words and quotes
- Search songs by the feelings and emotions expressed in it

Searching songs by artist, date, genre, album, and of course, name, are definitely crucial, but searching them by words or quotes that are present in the lyrics is something that we wanted to develop since the beginning as it is one of the biggest focus of our project and it's from there that we want to explore the data.

One of the objectives of this project is to implement a search system, so search queries are imperative. Being that, here follows the description of the planned search queries that may be more common:

7.1 Songs that Express a Feeling

An actor that uses our system should be able to search a song based on an emotion that it expresses. More specifically, by searching "Sad" (assuming that we are looking for sad songs), our system must be able to find songs with lyrics (or titles) that have the words "sad" or "cry" in it, and that, consequently, express the feeling of sorrow.

7.2 Songs with a Purpose

By "Purpose", we intend to refer to a context where the song fits in or an environment that can be created by it playing as a background sound. Clarifying, a user is able to search songs one can dance to or play at a party. In order to fulfill that, our system must search for songs that have "Dance Pop" as a genre or that have the words "party" or "shake" in the *Lyric* column.

7.3 Songs from specific Artists or Albums

This tool is very simple since it searches a song by its artist or album that are represented in the columns *Artist* and *Album* respectively. We feel that this is an important element since when we really like a song we tend to want to find others that are similar at some level so we search songs from the same album or by the same artist. Consequently, our system must be able to recognize that, by searching "Drake", "Drake" exists in the column *Artist*, or that by searching "I Am... Sasha Fierce", it exists in the column *Album*, and return all the songs associated with them. Even though we focused in artists and albums in this subsection, the songs can also be searched by *Title*, *Date* and *Genre* in the same way that they can be searched by *Artist* and *Album*.

7.4 Songs by Decade

Other tool that has interest is the search of songs based on the date they were released in. Even though the songs present in our dataset are only from the 2000's and the 2010's, something that would invigorate our system would be the selection based on chronology, not only the sort by decade, but also, for example, songs released in the year of 2011 or in the second half of the decade of the 2010's.

8 Collection Composition

With the use of *Solr*, we indexed our collection.

8.1 Song Document

The Collection only has one document, *Song*, as it is the only result our search system should retrieve. *Song* has, as mentioned before, *Title*, *Artist*, *Album*, *Lyrics*, *Top Genre* and *Date* fields, as well a new field *id*, added to aid the Evaluation process.

8.2 Indexing

For indexing, we used the POST tool provided by *Solr*, using a .csv file to populate the collection and a .json file for our schema. *Solr* has options for indexing and, after analysing that and the different data columns, we chose the following columns for indexing: *Title*, *Artist*, *Album*, *Lyrics*, *Top Genre* and *Date*. The Lyrics column contains large strings that allow text search.

8.3 Schema

Three field types were created on our schema: *Text*, used for *Title*, *Album*, *Artist* and *Top Genre* fields, *Date*, used only for the *Date* field and *LyricText*, used for the *Lyrics* field.

8.3.1 *Text*

This field type is used for the simpler text fields, which will never have many words and are mostly used as names. Taking this into account, we used the Standard Tokenizer, a simple tokenizer that separates the text on whitespaces and punctuation marks. When it comes to filters, we applied an Edge N-Gram filter, to subdivide each token into smaller tokens, as well as an ASCII folding filter and a lower case filter to standardize the text.

8.3.2 *Date*

The *Date* field type is used exclusively for the "Date" field, and it is a very simple field type, as the date can only give us so much information. This meant that all we did was use *Solr*'s *DateRangeField* class.

8.3.3 *LyricText*

The "Lyrics" field is the most important and more text rich field in our document. Hence, we applied a stopword filter, to remove stopwords, as well as the ASCII folding filter and the lower case filter to standardize the text. The Standard Tokenizer was used.

9 Search Systems

We chose 3 search systems to help our evaluation process.

9.1 System 1

For our first system, no schema is used and the search is made with a basic query, such as searching for a single word in the "Lyrics" field.

9.2 System 2

On the second system, our schema, described in the section above, is used. The search is still based on the basic query of searching for a single word in the lyrics.

9.3 System 3

For our third system, we also use our schema, but the search is made with an improved query, searching for more words in the "Lyrics" field, as well as searching for words in the "Title" or "Top Genre" fields, for example.

10 Query Execution

The Queries were executed in the different systems, so we analysed and compared the results. We used the *Lucene* parser.

We chose three queries to illustrate this:

10.1 Sad Songs

This first query is very simple, but also very common, as many people want to search for this topic and most relevant songs will probably contain the same words (e.g. "sad").

In the first and second systems, the query is translated into simply searching for the word "sad" in the "Lyrics" field. In the *Lucene* parser, this would be inserted as "Lyric:sad".

In System 3, there are improvements, as we add a search for "cry" in the "Title" field, also adding a multiplier to that search, as well as a search for the words "cry" and "sad" both appearing simultaneously in the "Lyrics" field, multiplying this even more than the search in the "Title" field. This is translated into the *Lucene* parser as [Title:sad^2 OR Lyric:sad OR (Lyric:cry AND Lyric:sad)^3]

10.2 Songs to dance in the shower

This query is slightly more subjective, as "danceability" is more tied to ryhtm than lyrics. Still, there are common words in songs for dancing. Moreover, we also have the genre of the song, which can help identify relevant results.

In the first two systems, there is a simple search for the word "dance" in the "Top Genre" field, translated as [Top_Genre:dance] in *Lucene*.

For the third system, besides repeating the same search as before, there is also a search for songs with the word "dance" in the genre and the words "dance" or "shake" in the lyrics, also adding a multiplier. In the *Lucene* parser, this is [Top_Genre:dance OR ((Top_Genre:dance Lyric:party)^2 OR (Top_Genre:dance Lyric:shake)^2)]

10.3 Nostalgic Songs

The last query is the most subjective one, as there are few to none mentions of the word "nostalgic" in nostalgic songs. Thus, we tried searching for the word "miss", although it could be misleading as that word has multiple meanings.

For Systems 1 and 2, the search was simply for "miss" in the lyrics, meaning we would input [Lyric:miss].

For the third system, we also searched for "miss" in the "Title" field, as well as for the words "loved" or "liked" in the lyrics. In the *Lucene* parser, this would be translated as [Title:miss^2 OR Lyric:miss OR Lyric:loved OR Lyric:liked]

11 Evaluation

With the indexing of the data done, we proceeded to evaluate the search system to guarantee that it fulfills our goals. It is important to analyse the ability to find the right information, and, in order to do that, we resorted to a standard IR system evaluation.

In our case, the query was first executed and the first ten results were analyzed to identify the relevant results. Precision is then calculated in a deterministic way, as described below, and we also calculated Average Precision.

Average Precision (AP) calculates the average of the precision values calculated every time there is a new relevant item.

This method meant that we could not calculate Recall, as we did not have every relevant item listed for each information need.

$$P = \frac{\#(relevant\_items\_retrieved)}{\#(retrieved\_items)}$$

In the equation above, P is the Precision. It will be used, alongside Average Precision (AP), to compare the performance of the different systems. We noticed that our schema produced the same results as an execution with the default schema, which is something to improve. It is also why we only present two precision-recall curves for each query.

11.1 Query 1: Sad Songs

- System 1 (No Schema)

  $P = 5/10 = 0.5$

  $AP = 0.564727$

- System 2 (Schema)

  $P = 5/10 = 0.5$

  $AP = 0.564727$

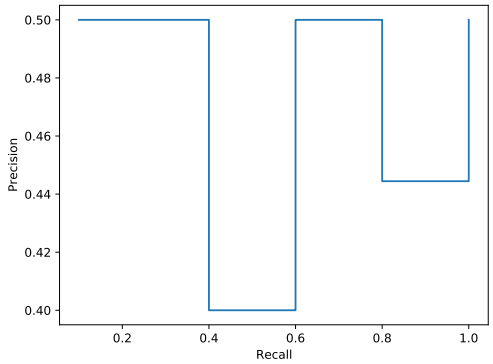

Figure 7: Precision-Recall curve for Query 1 - with and without schema.

- System 3 (Schema with Boost)

  $P = 6/10 = 0.6$

  $AP = 0.788183$

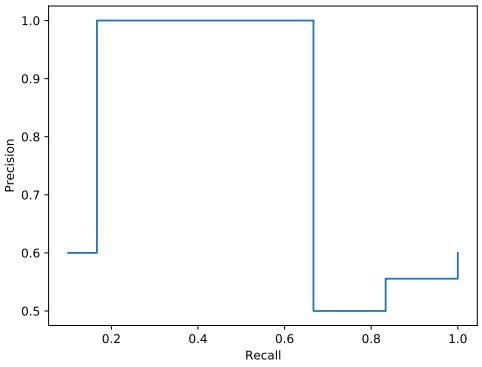

Figure 8: Precision-Recall curve for Query 1 - schema with boost.

In the first two systems the performance is similar with a precision of 0.5. In System 3 the precision is equal to 0.6. This higher value can be justified by the improvement the query introduces in this system, since it doesn't only take into account the presence of the word "sad" in the *Lyric*, but also values it even more, along with the word "cry" and it's presence in the *Title*, since they imply the feeling of sadness and are more likely to be mentioned in a "sad song".

11.2 Query 2: Songs to Dance in the Shower

- System 1 (No Schema)

  $P = 5/10 = 0.5$

  $AP = 0.372928$

- System 2 (Schema)

  $P = 5/10 = 0.5$

  $AP = 0.372928$

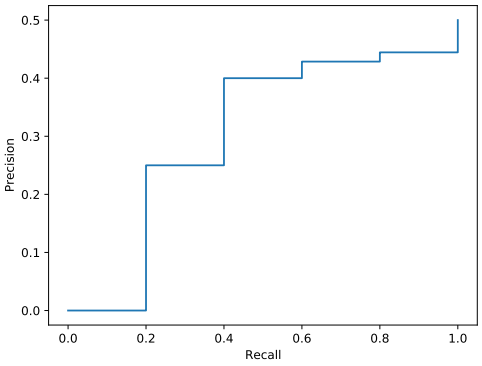

Figure 9: Precision-Recall curve for Query 1 - with and without schema.

- System 3 (Schema with Boost)

  $P = 7/10 = 0.7$

  $AP = 0.727381$

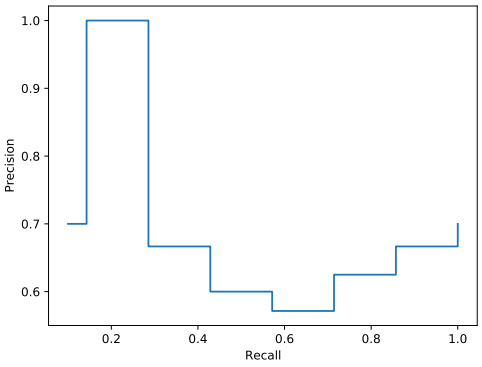

Figure 10: Precision-Recall curve for Query 1 - schema with boost.

Just like in the previous query, the first two systems don't retrieve all the relevant items from the set with a recall equal to 0,71, and retrieve irrelevant ones too, with a precision of 0.5. In System 3 the precision is equal to 0.7. As we can see, once again System 3 has higher values than the other two, which is also justified by the second-level query applied, that, besides taking into account whether the *Top_Genre* has "dance" in it or not, like the other two systems, also prizes the presence "shake" or "party" in the *Lyric*, which relate to dance and are more likely to be mentioned in "songs to dance in the shower". Additionally, while doing this query we realized that a possible improvement for the future was to keep data about the beats per minute (bpm) of a song, as it would be an important asset to better determine whether or not a song danceable.

11.3 Query 3: Nostalgic Songs

- System 1 (No Schema)

  $P = 8/10 = 0.8$

  $AP = 0.973765$

- System 2 (Schema)

  $P = 8/10 = 0.8$

  $AP = 0.973765$

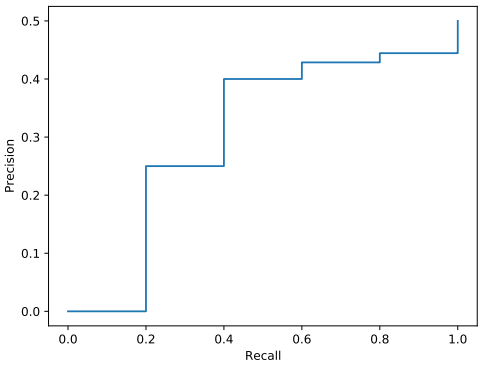

Figure 11: Precision-Recall curve for Query 1 - with and without schema.

- System 3 (Schema with Boost)

  $P = 7/10 = 0.7$

  $AP = 0.488183$

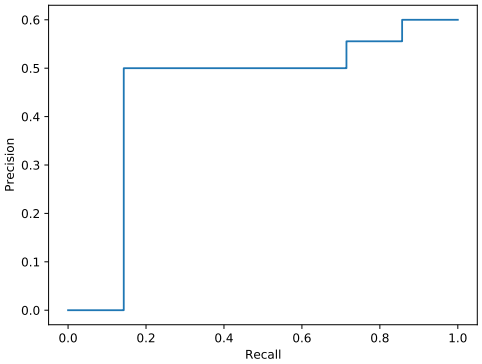

Figure 12: Precision-Recall curve for Query 1 - schema with boost.

In this query, for the first two systems the precision is equal to 0.8, which is a good value. In System 3 the precision is equal to 0.7, which is a slightly smaller that the precision of the other 2 systems. Contrary to the first two queries, in this one the System 3 has a lower precision value. There is a possibility that this is due to the fact that in our second-level query we take into account the presence of "loved" and "liked" in the *Lyric*, and, even though they are likely to be used in a context of nostalgia, they can be used in many other contexts, so the word "miss" is way more precise. The frequency of that word in the *Title* is also very uncommon, so *Title:miss^2* in the second-level query doesn't allow a much bigger improvement. In order to improve this query, the presence of "miss" in the *Lyric* should be even more valued, changing *Lyric:miss^2* to *Lyric:miss^3*.

12 Improvements

After the first two milestones, we proceeded to analyze our project and try to understand where we could upgrade some of its aspects. This way, we changed some parts relative to the first and second milestones with the goal to improve our information system. These improved parts we mentioned are described in the following subsections.

12.1 Dataset

The first big flaw we thought we could improve in our project was the dataset. We started with a dataset comprised of multiple artists' songs with the respective lyrics and decided we wanted to have each songs' genre. To achieve this, we joined other datasets with songs and their genre, but the overlap between our base dataset and the new datasets was minimal, which meant we had not much data. Furthermore, we felt like losing that much data just to have a new genre column wasn't the best way to go about our proceedings. We also realised we didn't have much information about each song. We felt that having more information about

each song, as well as having a better process to build a dataset would greatly improve our search system.

To improve this, we changed our method for obtaining data. We kept the original dataset with songs and lyrics, and compiled a Spotify playlist with every song from each artist in the dataset. We then used Spotify's API to collect information about each song, not only the genre but also interesting metrics that could possibly be used in the future, more concretely Duration, Popularity, Beats Per Minute, Danceability, Energy and Valence. This resulted in another dataset with every song from each artist and the genre and the metrics of each song.

We then merged the original dataset with the one just mentioned, resulting in a final dataset with the following information:

- *Artist:* name of the singer or band to whom the authorship of this song belongs to
- *Title:* title of the song
- *Top Genre:* genre of the song
- *Duration:* length of the song in seconds
- *Popularity:* popularity of the song, the value will be between 0 and 100, with 100 being the most popular
- *Beats Per Minute (BPM):* overall estimated tempo of the song
- *Energy:* measure of intensity and activity between 0.0 and 1.0
- *Valence:* measure from 0.0 to 1.0 describing the musical positiveness conveyed by the song

This improvement lead to an increase in the dimension of our final dataset, which has 1323 songs, more than double than the dataset described in the *Domain Conceptual Model* section.

12.2 Schema

When we first created our schema for analysing each field in our data entries, it was a very simple schema, with basic filters and tokenizers. This became clear when the evaluation was done, as there was no difference between searching with and without schema. We also realised we have a big text field, the *Lyrics* field, over which most of the search will be, has room to be better analyzed. Thus, we felt like improving our schema and exploring new ways to analyze each field would improve our search system.

We explored *Solr*'s documentation to find any filter we thought made sense in our case, and ended up adding a few to our schema.

12.2.1 Stopword Filter

We decided to add a stopword filter to the *Lyrics* field, to remove some words of our choice when querying over that field. In this case we used *Solr*'s ManagedStopFilter to remove the words "song" and "songs" from the query. This will allow the system to ignore those words and just search for the other words. For example, if there is a query such as "love songs", the word "songs" will be ignored and our system will search the *Lyrics* field for the word "love". We decided to keep every other word, as if a user searches for a songs exact lyrics, the frase should be kept in full.

12.2.2 KStem Filter

The KStem filter turns, for example, the words "jumping" and "jumped" into "jump". We concluded this would be interesting for our *Lyrics* field, to generalize some words. KStemFilterFactory from *Solr* was used.

12.2.3 EdgeNGram Filter

This filter was already in use in the *Title* field and we decided to expand it to the *Lyrics*, to generate smaller tokens of each word, broadening the search possibilities. We used *Solr*'s EdgeNGramFilterFactory, with a minimum gram size of 3 characters and a max gram size of 7.

12.3 Evaluation

For the evaluation of our improvements, we repeated the queries described in Sections 10 and 11, in order to obtain a true comparison of our search system before and after improvements. This meant we repeated the three search systems mentioned in Section 9 for each query.

Furthermore, calculated the same metrics, Precision at 10 (P) and Average Precision (AP). The actual results are also indicated by relevancy, with 0 representing a non-relevant result and 1 a relevant one.

12.3.1 Query 1: Sad Songs

- System 1 (No Schema)

  $Results$: 1010101101

  $P = 6/10 = 0.6$

  $AP = 0.613183$

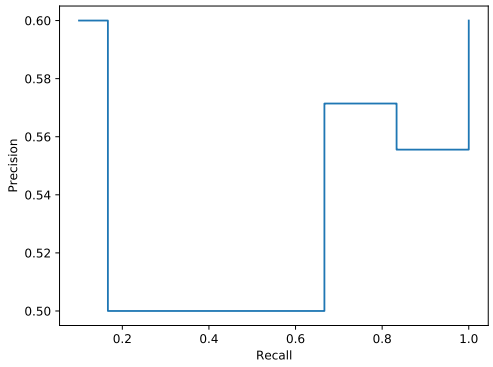

Figure 7: Precision-Recall curve for Query 1 after improvements - without schema.

- System 2 (Schema)

  Results : 1010101010

  $P = 5/10 = 0.5$

  $AP = 0.599295$

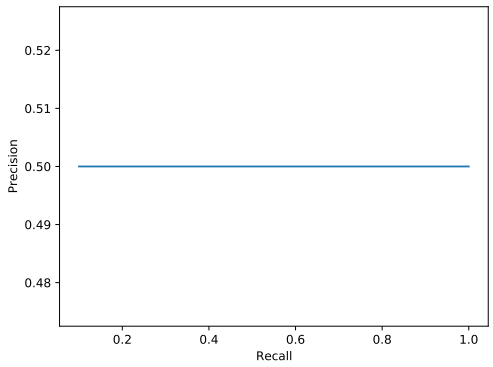

Figure 7: Precision-Recall curve for Query 1 after improvements - with schema.

- System 3 (Schema with Boost)

  Results : 0110011111

  $P = 7/10 = 0.7$

  $AP = 0.492196$

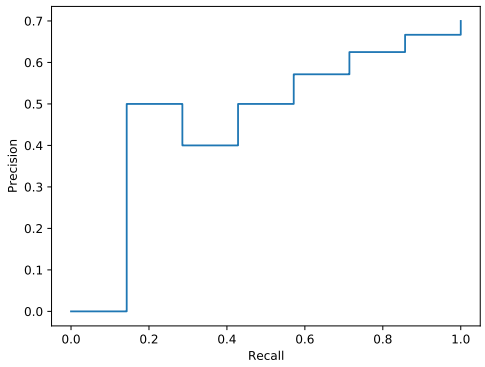

Figure 8: Precision-Recall curve for Query 1 after improvements - schema with boost.

Comparing with the results obtained before the improvements, these results are generally slightly better, with higher metric values overall. It is also of note that, after improvements, System 1 and 2 have different results, which was not the case before. In this case, System 2 has a worse Average Precision, but it yielded more results in total.

12.3.2 Query 2: Songs to Dance in the Shower

- System 1 (No Schema)

  Results : 1111001001

  $P = 6/10 = 0.6$

  $AP = 0.817945$

- System 2 (Schema)

  Results : 1111001001

  $P = 6/10 = 0.6$

  $AP = 0.817945$

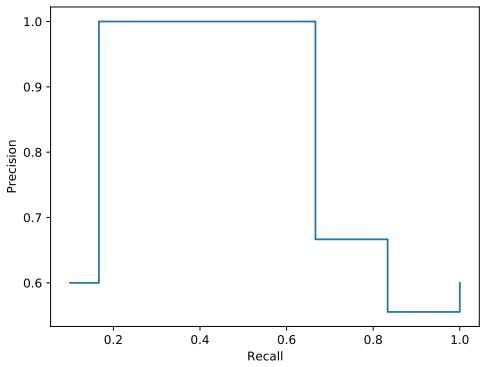

Figure 7: Precision-Recall curve for Query 2 after improvements - with and without schema.

- System 3 (Schema with Boost)

  Results : 1011111100

  $P = 7/10 = 0.7$

  $AP = 0.784436$

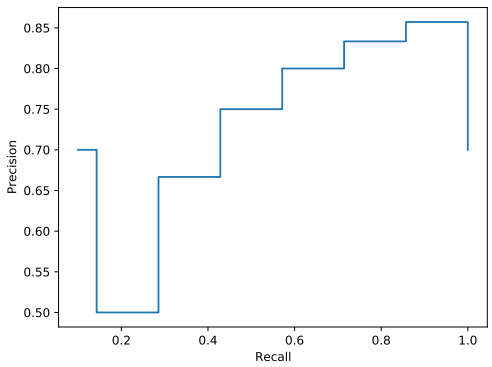

Figure 8: Precision-Recall curve for Query 2 after improvements - schema with boost.

In this case, there was no difference between having the schema or not. These results are also better in comparison to the ones before the improvements, with better Precision at 10 and Average Precision values in general.

12.3.3 Query 3: Nostalgic Songs

- System 1 (No Schema)
  Results : 1101010101
  $P = 6/10 = 0.6$
  $AP = 0.715035$

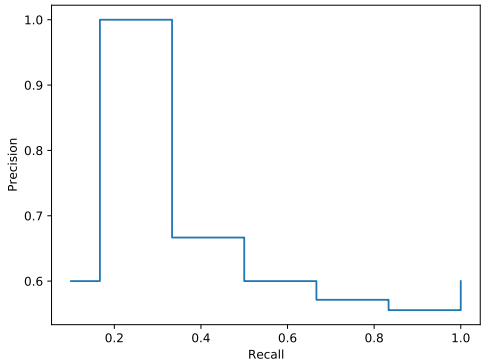

Figure 7: Precision-Recall curve for Query 3 after improvements - without schema.

- System 2 (Schema)
  Results : 1101111010
  $P = 7/10 = 0.7$
  $AP = 0.826102$

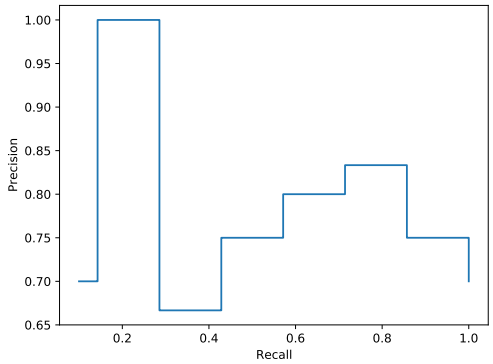

Figure 7: Precision-Recall curve for Query 3 after improvements - with schema.

- System 3 (Schema with Boost)
  Results : 0111110111
  $P = 6/10 = 0.8$
  $AP = 0.643563$

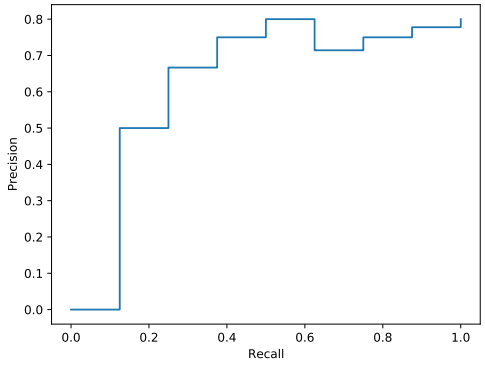

Figure 8: Precision-Recall curve for Query 3 after improvements - schema with boost.

For the last query, the results were generally worse than before the improvements, but, once again, Systems 1 and 2 have different results, with System 2 evaluating better.

12.4 Improvement Conclusions

To conclude, we are satisfied with our improvements, as we feel they resulted in a better search system. The results of the *Precision* and *Average Precision* that we had previously compared to the values we got now show that the search system improved, so the changes made to our work were adequate and beneficial.

13 Revisions

Since the report is an essential component to our project, being crucial to its understanding, goals and methods followed, we also tried to improve it and apply the suggestions and corrections made by the teacher. We feel that the grammatical and punctuation errors and redundant use of certain words are not worth mentioning in this section, even though some recommendations were extremely important to achieve the final report we have.

Even though initially the *Abstract* section demonstrated the purpose and theme of our project, it lacked the description of the goals and work covered. When we understood what the *Abstract* section should contain and approach, we added important information that made it relevant and adequate to anyone who wanted to in some type of way have a resume of our report, which is the point of it.

The *Introduction* section also suffered some changes since it could express more the relevance of our theme than it expressed in the report delivered in the second milestone.

In the *Conclusion* section, we decided to mention what was done taking into account what was originally proposed, which we consider its an important part that was not present before.

The *References* section used to be a numbered section, which was not supposed to happen, so we corrected it. We also added the last accessed date, which is the same for every website since we revisited all of them while finishing the report.

14 Conclusion

In the first phase of the project, we found and chose the data that felt relevant to our purpose. We combined it, refined it and came with an organized dataset that contains the information that we desire and that we intend to work with. The theme of music was a mutual interest, and the dataset felt interesting and relevant. The Data Model and the Pipeline were also crucial points to this project that were developed and

that helped us during the elaboration of the work until now since they allowed a better organization regarding tasks and work that needed to be done. The next step was mostly related to information retrieval and the search of songs. For this, *Solr* was crucial since it was a great help regarding the indexing of items. The creation of queries was also a fundamental part of the system as we revealed interest since the beginning in retrieving songs based on the feeling they convey and how we could make the process go beyond simply searching songs based on the *Title* or *Artist*. Finally, in the last phase of our project, we improved some aspects such as the dataset and the schema, with the intention to improve our search system. The process of *Evaluation*, which was also done before, allowed us to compare the quality of our work with the modified aspects to the work we had at the end of the second milestone. By analyzing the results, we could see that the improvements did in fact contribute to upgrade the search system. Nevertheless, an aspect of our work that we wish it would be better was the dataset, since the number of songs is not enormous, even though it's reasonable and we could get even more songs at the end. To conclude, this project allowed to improve and establish good knowledge basis regarding search systems and information retrieval. Looking back at our work, we feel that we fulfilled the goals established in the subject of PRI, and we worked with a software that can be very useful in our future, *Solr*.

References

[1] - "Genius (website)" Wikipedia, Wikipedia Foundation, accessed 19 january 2022, ⟨https://pt.wikipedia.org/wiki/Genius_(website)⟩

[2] - "Spotify" Wikipedia, Wikipedia Foundation, accessed 19 january 2022, ⟨https://en.wikipedia.org/wiki/Spotify⟩

[3] - "Song Lyrics Dataset" Deep Shah, Kaggle, accessed 19 january 2022, ⟨https://www.kaggle.com/deepshah16/song-lyrics-dataset⟩

[4] - "Web API", Spotify for Developers, accessed 19 january 2022, ⟨https://developer.spotify.com/documentation/web-api/⟩

[5] - "Spotify Past Decades Songs Attributes", Nicolas Carbone, Kaggle, accessed 19 january 2022, ⟨https://www.kaggle.com/cnic92/spotify-past-decades-songs-50s10s⟩

[6] - "Spotify - All Time Top 2000s Mega Dataset", Sumat Singh, Kaggle, accessed 19 january 2022, ⟨https://www.kaggle.com/iamsumat/spotify-top-2000s-mega-dataset⟩

[7] - "Spotify reveals the decade's most-streamed songs, from Ariana Grande to Drake" Mark Savage, BBC, accessed 19 january 2022,, ⟨https://www.bbc.com/news/entertainment-arts-50642141⟩

[8] - "Psychology of music preference", Wikipedia, Wikipedia Foundation, accessed 19 january 2022, ⟨https://en.wikipedia.org/wiki/Psychology_of_music_preference⟩

[9] - "Playlist Machinery", Playlist Machinery, Playlist Machinery, accessed 19 january 2022, ⟨http://www.playlistmachinery.com⟩

[10] - "Applying Data Mining for Sentiment Analysis in Music", Lucía Martín Gómez, María Navarro Cáceres (2017), Published in PAAMS, DOI:10.1007/978-3-319-61578-3_20, Corpus ID: 13987303

[11] - "Evaluation measures (information retrieval)", Wikipedia, Wikipedia Foundation, accessed 19 january 2022, ⟨https://en.wikipedia.org/wiki/Evaluation_measures_(information_retrieval)⟩

[12] - "Getting Started with Apache Solr", Sematext, Sematext, accessed 19 january 2022, ⟨https://sematext.com/guides/solr/⟩

[13] - "Welcome to Apache Lucene", Apache Software, The Apache Software Foundation, accessed 19 january 2022, ⟨https://lucene.apache.org⟩

[13] - "Spotify", Apache Software, The Apache Software Foundation, accessed 19 january 2022, ⟨https://www.spotify.com/⟩