500 Best Albums Ever: Search System

Eunice Amorim, Henrique Sousa, Henrique Nunes

Abstract

This report focuses on the creation of an Information Retrieval system about the 500 best music albums ever produced according to *Rolling Stone* magazine. Thanks to *Wikipedia*, *Spotify* and *Genius* it was possible to assemble the underlying data set with relevant attributes to identify and search for a song. To achieve this, a pipeline was built considering 3 main steps: collecting, processing and analyzing. As a result of this work, a final data set of 9.21 MiB was gathered. Some interesting findings about the songs collected include: average duration between 3 and 5 minutes and higher occurrence of older songs. This data set was indexed using the *Solr* engine and a set of default filters. They allowed to search for songs according to the established information needs resulting on a Mean Average Precision of circa 0.67 in this process. Additionally, the possibility of using other sets of albums was taken into account. The methods used allow for the addition of new albums and albums in different languages.

I. Introduction

Over the years, many tried to discover the best music albums ever released. One of these initiatives is the annual listing of the magazine *Rolling Stone*: the 500 best albums ever released [Mag20]. One reading this list may want to know more about each album, discover how many songs they have, what songs are included, and how good they are. For this reason, this project intends to build a search system which allows the user to search over this ranking with different metrics (album, artist, lyrics, year, ...) and thus satisfying its curiosity. Since there are many editions of this list, the focus was on only one: the 2020 version [1].

This paper aims to describe the procedures performed in order to build an Information Retrieval system based on referred list. It is structured according to the sequence of steps performed, namely Data Preparation, Information Retrieval and Search System. The Information Retrieval step consists on collecting and indexing the data, retrieve documents and evaluate the results while the Search System is an improvement over the one built on the previous step.

II. M1 - Data Preparation

A. Data Sources

After some search, an already assembled data set with all information wanted (list of albums, artist, songs of each album) was not found. For this reason, it was decided to build a customized data set using the following sources:

A.1) Wikipedia

Wikipedia[2] is a platform of shared knowledge [Wik22]. It was hard to extract all albums from the original magazine article because all albums had a different URL. So, since *Wikipedia* provides the complete list [3] in a easier way to extract, it was decided to use it. In order to be sure about the validity of the list, coherence between *Wikipedia* and *Rolling Stone* was manually checked being confirmed as a reliable source. Regarding the licensing, the list is property of *Rolling Stones* magazine and is publicly available.

A.2) Spotify

Spotify[4] is a music streaming website used by millions all over the world. It contains 80 million tracks being able to discover the information of all albums (7,112 songs). The data is publicly accessible [Spo22a] and usable for this kind of project [Spo22b].

A.3) Genius

One of the main lyrics websites is *Genius* [5] . It has a community of 2 million contributors [Gen09] including verified artists [Gen22c]. As a result, this was a natural choice to gather this information. It proved to be effective, only missing 738 song lyrics (check the Section Collecting lyrics for more details). The song lyrics are publicly accessible and usable for this project [Gen22b] but belong to the authors of the song.

A.4) Alternatives

Other options were considered and tested. The most relevant one was *Musixmatch*. Due to the limitations of the *Musixmatch* free package, it was not used.

B. Data Collection and Preparation

With the data sources chosen, the next step was gathering the data from said albums in a way that would fit the vision for the search engine. To assemble the data set, the name of the song, the album, the artist, and the lyrics would be imperative to be obtained. Additional metrics would also end up being collected, like the duration of each song, its release year and its album ranking.

B.1) Pipeline

During the development processes several hurdles had to be overcome, mainly when it comes to the quality of the data.

[1] https://www.rollingstone.com/music/music-lists/best-albums-of-all-time-1062063/

[2] https://en.wikipedia.org/wiki/Main_Page

[3] https://en.wikipedia.org/wiki/Wikipedia:WikiProject_Albums/500

[4] https://open.spotify.com

[5] https://genius.com/

After several iterations it was settled for the pipeline observed in Fig. 1. Each main pipeline component is explained in the following sections.

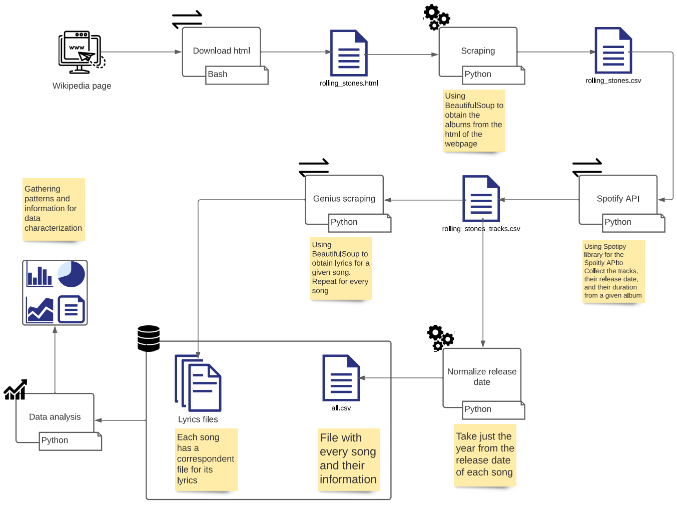
Wikipedia page
Download html
Bash
rolling_stones.html
Scraping
Python
rolling_stones.csv
Using BeautifulSoup to obtain the albums from the html of the webpage
Gathering patterns and information for data characterization
Genius scraping
Python
rolling_stones_tracks.csv
Spotify API
Python
Using BeautifulSoup to obtain lyrics for a given song. Repeat for every song
Using Spotipy library for the Spoitiy APIto Collect the tracks, their release date, and their duration from a given album
Normalize release date
Python
Lyrics files
all.csv
Data analysis
Python
Each song has a correspondent file for its lyrics
File with every song and their information
Take just the year from the release date of each song

Fig. 1: Pipeline to collect and process data

B.2) Collecting albums

The entry point input is the *Wikipedia* list of the 500 best albums according to *Rolling Stone* [Wik20], which replicates the one in the magazine's website [Mag20]. However, since there is a single table, the *Wikipedia* page was found to be easier to scrape. The webpage is first stored locally. Afterwards, a python scraper collects the artists and the albums, sorted by ranking order, and stores them in a file.

B.3) Collecting songs

Next, the file obtained is iterated in the command line, taking the album and artist from each line, which is given as input to a python script that will collect every song of the given album. To do that, the *Spotify API* is used through the *Spotipy* library [Lam14]. The two tools will be addressed as a unit called either *Spotify* or *Spotify API* henceforth. The *Spotify API* allows to search for an album. This process was not trivial and metrics had to be developed to obtain the closest match (by *Levenshtein* distance). For example, remastered or deluxe edition might be the only versions available in *Spotify* for a given album, but there might have been an album with the same name from a different artist. After the album is chosen, its tracks are collected, alongside the song duration and the album release date. The script will append all of the relevant information to a single file that will keep the songs featured in the data set. Additionally, the album ranking is also present for each song.

B.4) Collecting lyrics

Finally, lyrics were retrieved from the songs obtained. The lyrics are collected from the *Genius* website [Gen22a]. The file with the tracks will be iterated line by line in the command line to obtain a single tracks' name and artist. The two are put together into a URL which is used to directly access the webpage of the song. If no result is found, the most likely culprit is the name obtained from the *Spotify's API* which is not always canonical. In that case, the name is sanitized and another attempt is performed. If one of the processes is successful, the program will scrape the HTML for the lyrics and the string will become the name of the file where each of the lyrics is stored. This string will also be added to the file which agglomerates the whole collection of songs so that referencing is possible.

Despite a *Genius API* being available, it wasn't reliable, leading to more missing information than the scraping did.

B.5) Processing

The data set at the end of the collecting process didn't need much processing since such processing was done while collecting. It would be impossible to gather a significant part of the data without sanitizing the song name for example.

However, a separated processing routine was viable: the normalization of the release date. The file obtained previously will be fed to a program whose function is to normalize the song's release date, since the format the *Spotify API* yields is not homogeneous. All of the dates are, therefore, standardized to include just the year of the release.

B.6) Makefile

To automate the process, a makefile was created to generate the different files. It is structured between the collecting, the processing and the analysis phases. The main file is structured in CSV format and each lyrics is contained in a TXT format.

C. Data Characterization

The assembled data set has been tailored to the needs of the project. As the sources used are reliable and reputed, it's reasonable to assume the correctness of it. Nevertheless, there are still conclusions that can be made.

C.1) Domain model

Data is made of two parts: the song data and the lyrics data. The song data is a single file with all of the songs available in the data set. It contains the name of the song, the album, the artist, the release year, the duration and the ranking of the album in which it is featured. The lyrics data is composed of one file per song and contains the lyrics for that song.

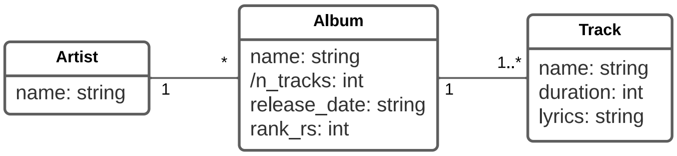

Fig. 2: Domain Model

C.1.1) Data fields

The name and lyrics of the track, the name of the album and the name of the artist are all string values. The duration of the track is an integer value with the number of seconds of the song. The date of the album is a integer value with the year of the release date of the album. The rank_rs is the *Rolling Stone* ranking of the album. The n_tracks is a derived attribute with the number of songs in the album.

C.1.2) Missing values

The only missing value are the lyrics for some of the songs. Due to some technical difficulties, gathering lyrics for all tracks was not possible.

C.1.3) Size and volume

The total data has a volume of approximately 9.21 MB. It contains a total of 7,112 tracks, 6,374 have lyrics. However, some of these lyrics are present in multiple albums and share the same lyrics' file.

C.2) Descriptive and exploratory statistics

With large data sets like this one, it is important to know the data in order to take advantage of its properties. Therefore, an exploratory analysis was performed. Plots were used to explore its contents and discover some patterns that allow a better understanding of it.

The album duration and the album mean track duration follow a similar distribution (Fig. A.1 and Fig. A.2). The outliers, especially, are similar. However, it cannot be concluded that this is due to the same album, since a long mean song duration could just imply that an album is made of a single long song, for example. It is possible to conclude that most albums are about 50 minutes in length and tracks are mostly under 4 minutes long which is consistent to the recommended 3-5 min duration [L19].

The majority of the albums have less than 20 songs. There are a few with more than that and some outliers with even more than 40 songs (Fig. A.3). It could be the case that the albums with this many songs are a result of an extended version of the album found by *Spotify*.

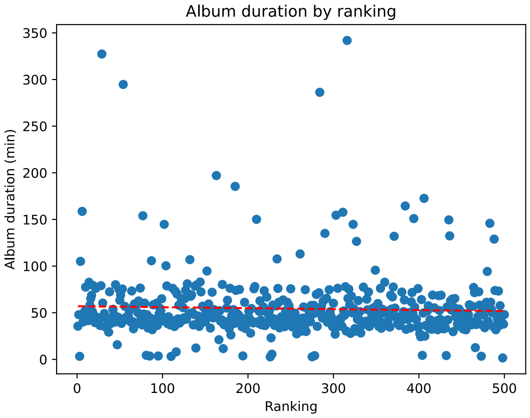

Fig. 3: Album duration by ranking

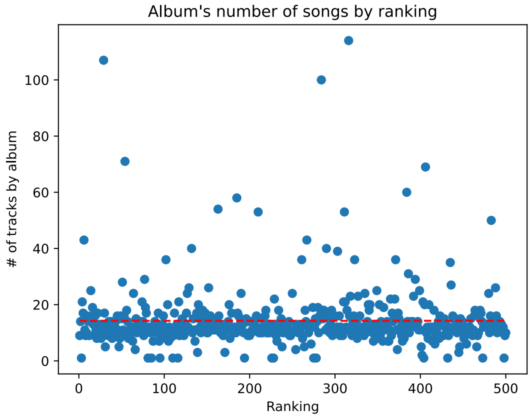

Fig. 4: Album number of songs by ranking

By plotting the linear regression that expresses the correlation between the above metrics and the ranking of the respective album, there are some conclusions to be made. Longer albums and albums with a higher track mean duration have a better ranking (Fig. 3 and Fig. A.4). It is impossible to determine if this is a causal relationship and, even if it was so, it would only make the album perform marginally better. Additionally, the number of tracks of an album appears to be mostly irrelevant to its ranking (Fig. 4).

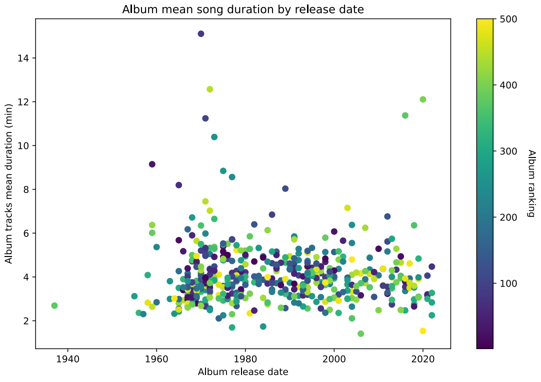

Fig. 5: Album ranking distributed by release date and mean track duration

The graphic in Fig. 5 shows the distribution of albums according to its release date and the mean time duration of its tracks. The markers in the plot are colored following a scale given from album ranking. Analysing the plot, it is possible to conclude that the best albums of the 500 are more agglomerated between the 60's and 90's and between 3 and 5 minutes per track, which was expected.

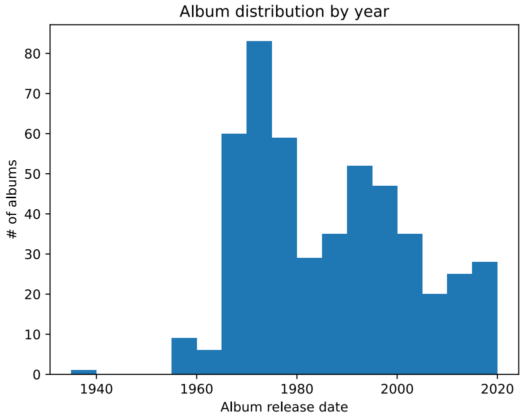

Fig. 6: Album distribution by year

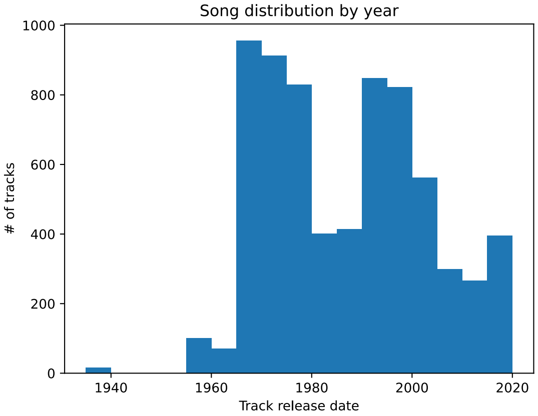

Fig. 7: Song distribution by year

The distributions for the album release date (Fig. 6) and for the track release date (Fig. 7) follow a similar distribution. This is to be expected, as more albums in a given year means, generally, more songs for that same year.

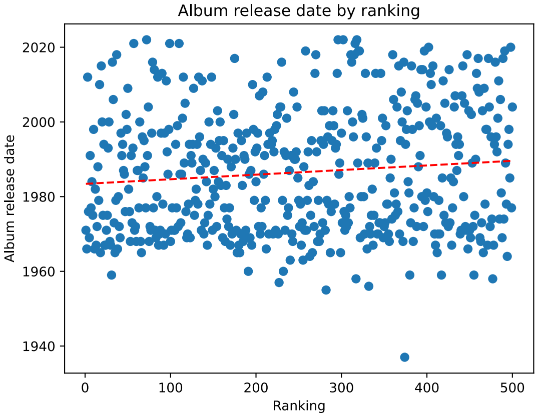

Fig. 8: Album release date by ranking

The graph in Fig. 8 plots the linear regression correlating the ranking of an album and its release date. Though small, it can be seen that an older album is associated with a better ranking. One hypothesis that explains this behavior is the nostalgia factor.

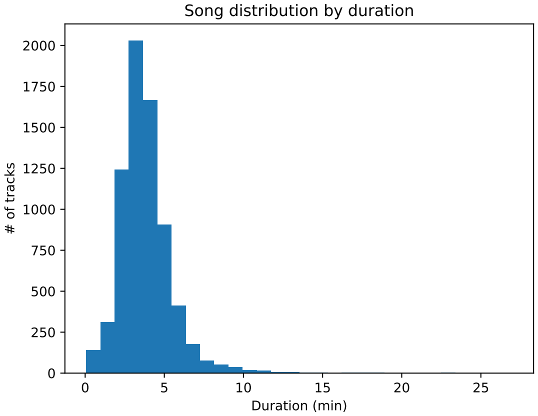

Fig. 9: Song distribution by duration

The song distribution by duration (Fig. 9) is similar to the album distribution by mean track duration. It is not surprising that the grouped data replicates the pattern of the individual data. As it can be seen, most songs have a duration of less than 5 minutes.

The graphic in Fig. A.5 presents an estimate of the number of tracks for which lyrics have been found. Despite being quite large in absolute number, the number of tracks for which the lyrics were not obtained is significant, but not crippling in terms of the effectiveness of the data.

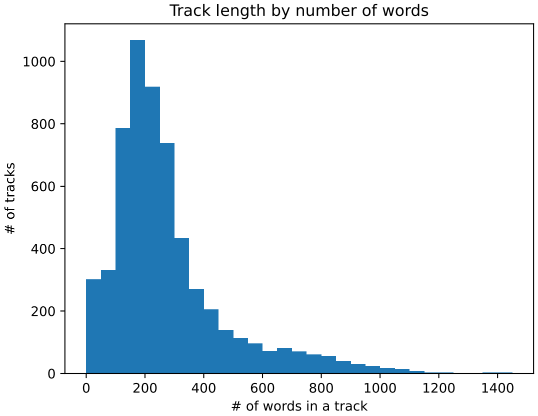

Fig. 10: Number of words by track

Most tracks tend to have a number of words smaller than 400 (Fig. 10). In Fig. 11, it is also possible to see that a considerable number of tracks, about 200, have no words at all. Those are instrumental only tracks. It was decided to keep them since they are part of the albums and it's still possible to work with its titles, year of release, duration, artist and ranking.

In Fig. A.6, a visualization of the frequency of the words in all songs can be seen, where a greater word size indicates a greater frequency. The words ”know”, ”love”, ”now”, ”got”, and ”want” have the most relevance.

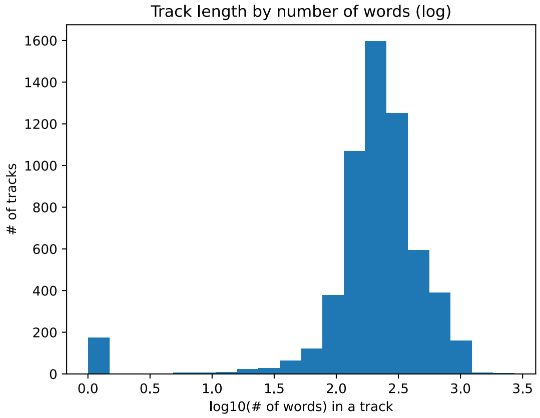

Fig. 11: Number of words by track (logarithmic scale)

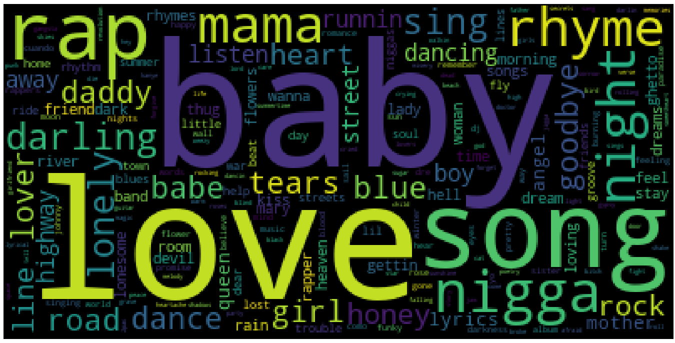

Fig. 12: Keywords

Above, in Fig. 12, a visualization of the keywords can be seen. Unlike the previous graphic, it evaluates only the 3 keywords in a track before calculating the relative frequency of each. It can be seen that the word ”love” continues to have a great relevance but some of the other predominant words, mainly common verbs, are not deemed to be a keyword.

D. *Prospective Search Tasks*

With all the information collected, it is important to understand which queries are relevant for the system. The final search system should be able to help users fulfilling their information needs. Some of them may include:

1) I want 1991 Nirvana songs having between 2 and 3 minutes
2) I want songs with a regretting tone
3) I want an underrated relaxing song
4) I want a song that speaks of love in a depressing way
5) I want a song that talks of life and love
6) I want songs about surprise and happiness
7) I want a song with a sentence like ”I like her”
8) I want the very best songs

In section M2 - Information Retrieval these search tasks will be translated to queries to retrieve documents from the system.

III. M2 - Information Retrieval

A. *Collection and Indexing*

After the data set has been built and analysed, and the information needs has been defined, the next step was building a system that contained a collection of indexed documents. These documents should represent the data collected in Section II and its indexing must be useful on a later task.

A.1) Choice of the tool

There are several search engines available to building the indexes, store data and querying it. Some of those are, for example, *Solr* [Sol22] and *Elasticsearch* [Sea22]. These prebuilt systems allow its users to access the system UI and easily query the system, finding out relevant documents and evaluate the system indexing. To this project, the choice fell on *Solr* because the team had more knowledge about it, which made the development faster.

A.2) Document definition

With the tool chosen, it was needed to define what would be a document in this system. For any given track, given that a small amount of data has a relationship to it, instead of using other cores or tables (introducing the need to merge operations and adding more complexity to the system), a document is comprised of the full data.

As such, a document is represented by Listing 1:

```
{
    "id": integer,
    "artist": string,
    "album": string,
    "album_release_date": integer,
    "album_ranking": integer,
    "n_tracks": integer,
    "track": string,
    "track_duration": integer,
    "lyrics": string
}
```

Listing 1: Document representation

In order to upload the data already collected to the Information Retrieval tool, it was firstly transformed into a JSON format, being all of the fields indexable.

A.3) Indexing

The default indexing provided by *Solr* was able to correctly index the data. However, it yielded a poor representation as every data field was made of an array with the data. For this reason, the first step taken was to make the fields into an atomic data type: a string or an integer.

For the integers, this was the only indexing performed as there was no need for a more complex approach.

As for the strings, a different approach was used. To all of the string fields, a standard tokenizer was used and different filters were applied according to its importance. For the artist, album and track a simple indexing was performed with ASCII folding and conversion to lower case. For the lyrics, more filters were applied to increase the efficacy of the system. Besides the filters applied to the title, additional ones were used to account for plurality, the English language's possessive case, English stemming and synonyms. The first three added were used to treat similar words as a single one (e.g. "loves", "love's" and "loving" would be transformed into "love"). After that, it's applied a filter to remove duplicates originated in the last ones. At the end, synonyms of the indexed words are also indexed so that a search can be more embracing.

A.4) Schema

The final schema can be represented as the following:

Field types

- textType
  - tokenizer
    - StandardTokenizerFactory
  - filters
    - ASCIIFoldingFilterFactory
    - LowerCaseFilterFactory
- lyricsType
  - tokenizer
    - StandardTokenizerFactory
  - filters
    - ASCIIFoldingFilterFactory
    - LowerCaseFilterFactory
    - EnglishMinimalStemFilterFactory
    - EnglishPossessiveFilterFactory
    - PorterStemFilterFactory
    - RemoveDuplicatesTokenFilterFactory
    - ManagedSynonymGraphFilterFactory
    - FlattenGraphFilterFactory

Fields

- artist: textType
- album: textType
- album_release_date: pint
- album_ranking: pint
- n_tracks: pint
- track: textType
- track_duration: pint
- lyrics: lyricsType

B. Retrieval

With the system built, it was possible to start implementing its main purpose, the document retrieval. The system should retrieve documents according to a specific query. Therefore, the information needs were converted to queries so one could request the system and get documents.

B.1) Retrieval process

The *Solr* project has already a friendly UI so that one can query the information system and retrieve some documents easily. With that in mind, after the information needs are defined, they were all translated to queries that can be placed in the *Solr* user interface. The queries used to test the schema are distinct in order to explore the different search properties of the system. The detail of each query can be found in Table A.1.

B.2) Fields boosts

In an information system, usually the data is distributed by several fields with different levels of importance according to the search task to perform. Therefore, in some queries it's good to boost some fields in order to attend the relevant documents with more effectiveness. An example of this boost was used in the query that translates the information need "I want songs with a regretting tone". To this query, the name of song was considered more important to have a match than the lyrics.

B.3) Term boosts

In a textual query, one may want to assign more weight to some terms of the query than others. In *Solr* search engine, this can be accomplished by adding the caret character, followed by the numeric weight assigned to the term right before the special character. As an example of this use case, relatively to the information need "I want a song that speaks of love in a depressing way", a query with the text "love^5 -good bad -happy sad" was made. This query had the goal to emphasise the word "love" guarantying that the documents retrieved were about love, but at the same time removing terms like "good" and "happy" and matching terms like "bad" and "sad". This accomplishes the depressing part of the information need.

B.4) Independent boosts

Another search boost that could be implemented is the Independent Boost. It allows to emphasise terms regardless of the query. To explore it, the boost could be used in same way of the Term Boost (by using the caret character after a specific term but in the DisMax "bq" field). However, another approach was followed. The DisMax "bf" field was applied to boost the "album_ranking", regardless of the query. This technique can be seen in query of the information need "I want an underrated relaxing song" by using the function "field(album_ranking)" which assigns priority to the albums with a worst ranking.

B.5) Phrase match with slop

In some cases it is wanted to search by phrase match. Nevertheless, this approach is very restrict as it normally retrieves a small number of documents. To turn the phrase match softer, it can be used with slop. The slop does not restrict the phrase terms to a specific order or consecutive position. Depending on its value, the search can be more or less embracing. This process was applied to the information need "I want a song with a sentence like "I like her" to search for instances of documents that contains a similar phrase to "I like her".

B.6) Wildcards and fuzziness

Wildcards and Fuzziness are useful to generalize some searches. They allow to match more terms from a single one. Fuzziness allows to ignore, for example, mismatches caused by typos and Wildcards allows to search for a term without exactly specifying it. The information need "I want songs about surprise and happiness" was translated to a query with wildcards like "surpr*" and "happ*" in order to match terms relative to "surprise" and "happiness", respectively, such as "surprisingly" and "happy".

B.7) Proximity search

According to Solr documentation [6], Proximity Search in the "lucene" parser refers to searches looking for terms within a specific distance from one another. This is very similar to a query phrase match with slop, but it is specified like an expression followed by a tilde character and a number that specifies the maximum proximity required. For instance, this proximity search was used on the query of information need "I want a song that talks of life and love" by querying the expression "life love" with a proximity of 3.

C. Evaluation

The system retrieves documents according to a query, however the relevance of the documents retrieved can vary. Therefore, it's important to evaluate the system by analysing the relevance of its results and verify if the system has a better performance than simpler ones.

C.1) Different setups

A set of different setups were used to index the data collected and evaluate the system.

- Schema 1 Simply sanitize the text
  - ASCIIFoldingFilterFactory
- Schema 2 Sanitize the text and convert to lower case
  - ASCIIFoldingFilterFactory
  - LowerCaseFilterFactory

These differed in the schema used. The final schema (Section A.4)) refers to the latest iteration (an increment of the previous by adding new filters to the lyricsType).

C.2) Manual evaluation process

In order to perform a good evaluation, the relevant documents for a given query need to be known. A manual evaluation was necessary to gather this data with the most certainty. However, obtaining this set from the thousands of documents is not easy and would be almost impossible through brute force. As such, approximations were made. The proposed query was ran and the results limited to 30. These were then manually evaluated and the totality of the relevant documents fetched from this set. This same query, using these schemas and others, will be evaluated according to the relevant documents that were found through this process. The relevant documents manually evaluated are registered in Table A.2.

C.3) Precision metrics

Precision metrics are a way of evaluating the information retrieval system. It measures the ratio of the relevant retrieved documents. It is important to know whether or not the document handed back to the user contained the information that they were looking for.

Query	Average Precision	Precision at 10 (P@10)
1	1.0000	1.000
2	0.5954	0.400
3	0.6523	0.600
4	0.6493	0.800
5	0.8537	0.800
6	0.6531	0.700
7	0.8556	0.400
8	1.0000	0.900

Table I: P@10 and Average Precision for each query

C.3.1) P@10 and Average Precision

Every query was evaluated with Precision at 10 and Average Precision metrics. The Average Precision was calculated based on the first 30 documents retrieved by the query (the query 8 is an exception to the rule since that it retrieved only 9 documents). Its results are represented in table I.

C.3.2) Mean average precision

Mean average precision is another precision metric, but that evaluates the Information Retrieval system on a global basis. In table II, the metric is presented not only for the final schema, but also for the other ones that have been developed proving the improvement throughout the iterations.

Schema	Mean Average Precision
1	0.6250
2	0.6875
Final	0.7000

Table II: Mean Average Precision for each schema

C.4) P-R curve

A P-R curve is a graph that tracks the precision of a query as the recall increases. This allows to characterize the compromise between both metrics and better make a decision on how many results to display in a system deployed to production.

Two of the obtained curves have a precision of 1 (Fig. 13, Fig. 14). This could be attributed to the fact that these queries do a simply matching in some of the attributes, yielding results with absolute certainty.

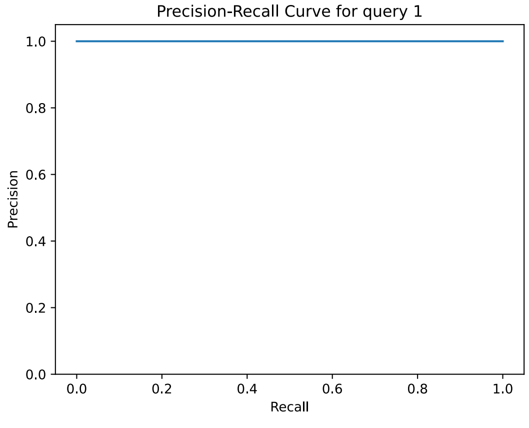

Fig. 13: Query 1 P-R Curve: Milestone 2

[6] https://solr.apache.org/guide/8_10/the-standard-query-parser.html#proximity-searches

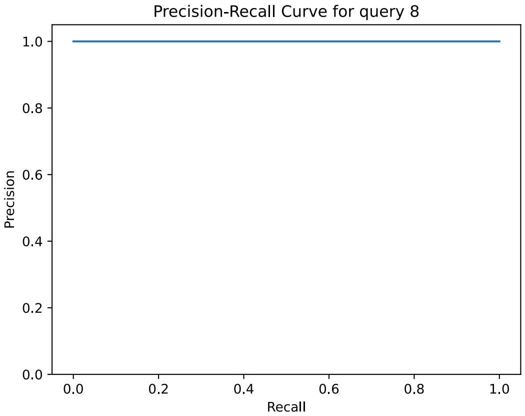

Fig. 14: Query 8 P-R Curve: Milestone 2

Another set of the curves display a monotonic decrease of the precision as recall increases, starting in 1.0 (Fig. 15, Fig. 16, Fig. 17). These indicate that the search system was able to correctly identify the first relevant documents, but struggles doing so afterwards.

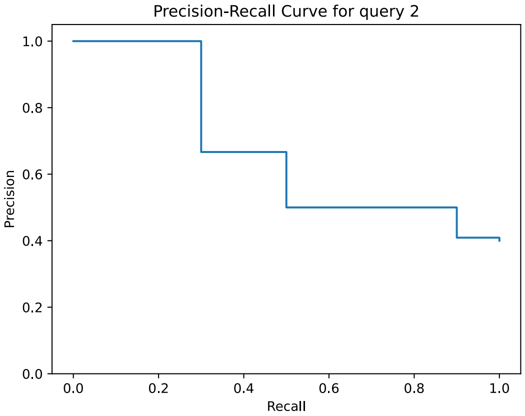

Fig. 15: Query 2 P-R Curve: Milestone 2

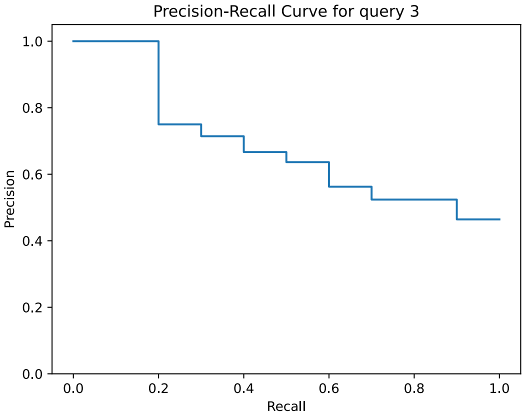

Fig. 16: Query 3 P-R Curve: Milestone 2

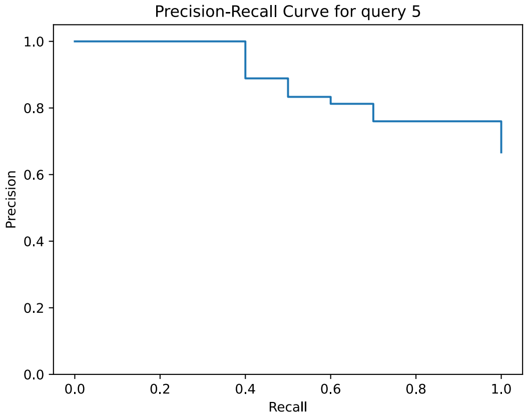

Fig. 17: Query 5 P-R Curve: Milestone 2

Another set has the same motonic decrease as the last one, but having the maximum precision under 1.0 (Fig. 18, Fig. 19). This could be indicative of finding a set of relevant documents after just a few non-relevant ones. The remaining could be said to have a mix of relevant and non-relevant documents, though the former are the most abundant, allowing the curve to not decrease a lot.

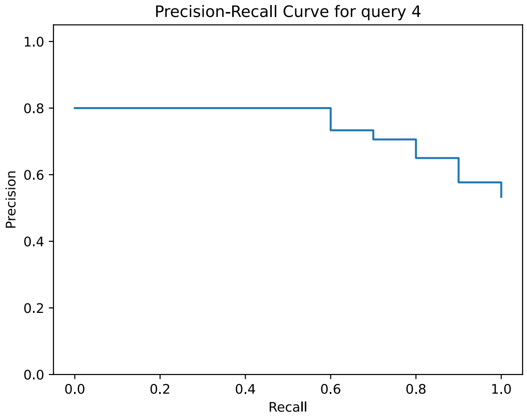

Fig. 18: Query 4 P-R Curve: Milestone 2

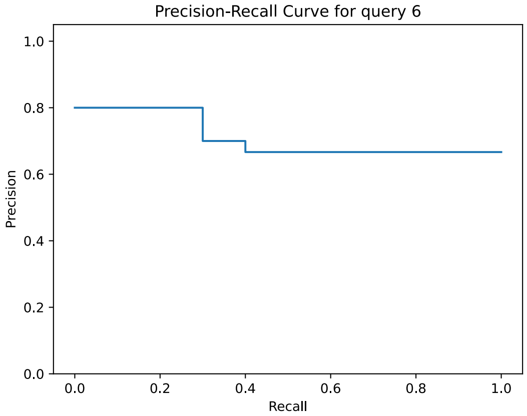

Fig. 19: Query 6 P-R Curve: Milestone 2

As for the last query (Fig. 20), the odd curve is explained by the reduced amount of relevant documents. The search system rapidly finds every relevant document but one, which is only found after many non-relevant ones were added to the retrieved documents.

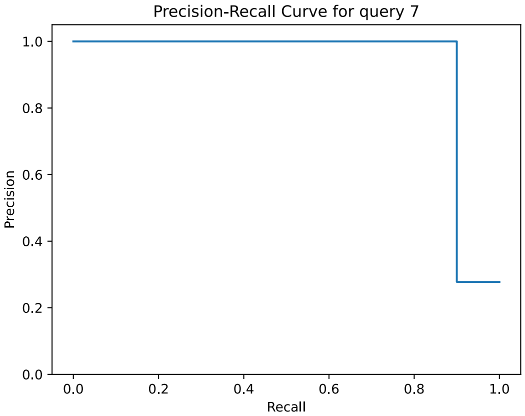

Fig. 20: Query 7 P-R Curve: Milestone 2

C.4.1) Discussion

Even though the metrics used to evaluate the system are adequate and used in the field, they have without doubt yielded skewed results, namely when it comes to recall related metrics. This is due to the manual evaluation process. The relevant documents obtained are not guaranteed to be the only ones that fulfill the information. Additionally, the process was toilsome and took a reasonable amount of time, diminishing the possibility of rapidly performing evaluation on different queries to determine improvements between widely different schemas. Another detail to consider is the limits of the system. The information need had to be transformed to abide by the search system's syntax, which reduced its expressiveness. A more complex search system would allow for a query in natural language. The domain and the data collected is also a limiting factor, for expressing feelings or types of musics using the lyrics as a main factor is not trivial.

IV. M3 - Search System

A. Improvement ideas

From the previous milestones, there were some aspects with inaccuracies. The more critical ones were the missing songs and dealing with songs in another language than English. For that reason, it was decided to fix this issues in the project's final iteration.

B. Data collection improvement

In order to achieve the final goal of building an accurate information retrieval system, the data collection process needed to be refined.

B.1) Spanish songs

The *Rolling Stone* list of albums contained a lot of different types of music. The most common language among them is English, however it was discovered one album with its lyrics in Spanish. Without a further treatment, this would lead to less accurate results as some of the *Solr* steps are language dependent. For example, the removal of English stop words could erase Spanish words that were not stop words.

Unfortunately, only with one album, it would be difficult to proper evaluate the improvements added. Therefore, it was decided to add some more Latin albums collecting a bigger set of those songs. With that, the effects of the treatment would be more noticed. As long as the discussion is about the best albums ever, a search was made in order to find the best Latin albums ever.

B.1.1) Data Sources

There are some other institutions that produce album rankings. One of those is the important magazine *Billboard* [Sta11]. One of those rankings is *The 50 Greatest Latin Albums of the Past 50 Years* [Sta15]. Given the reputation of this magazine, it was decided to add these albums to the initial ones. The considerations about data licensing of *Rolling Stone* apply here since it is a magazine article (publicly available and usable). However, this website revealed difficult to scrape. In alternative, it was found another one with the complete list where the process was easier: *List Challenges* [7]. A manual verification of correctness against the original list was made to use it with more certainty.

B.1.2) Pipeline

In order to add the new albums to the current dataset, a process similar to the original was employed. First off, a web scraper was used to fetch the albums and their ranking in a format compatible with the one used for the 500 albums. Next, the whole process of fetching the track names from *Spotify*, their lyrics from *Genius* and standardizing the release year was replicated making use of the already existing Makefile. This lead to the creation of two identical files which were therefore concatenated, yielding a single one with the new data assimilated.

B.1.3) Integration with Search System

Regarding specific language filters to correctly synergize with the different lyrics, *Solr* [Sol22] must be given the right language. Native language identification support was not able to be used. As such, *Spacy* [Spa22] was used to identify it. In accordence with the language, the data fed to *Solr* would have a different field containing the lyrics, making it possible to deal with the different languages in a different way.

B.2) Missing lyrics

As stated on Section Size and volume, there is a non-negligible number of songs whose lyrics are missing. Great effort was put into reducing this number.

B.2.1) Pipeline

Complementary to the pipeline already developed for the previous deliveries, another script was written whose objective was to fill in the remaining missing lyrics. The API *lyrics-extractor* [Agr22] was used for that effect. It provided a simple and more streamlined method to search for lyrics

[7] https://www.listchallenges.com/billboard-the-50-greatest-latin-albums-of-the

from a selection of handpicked websites. The tool recommend websites were used:

1) *Genius* [Gen22a]
2) *Lyricsted* [Lyr22d]
3) *LyricsBell* [Lyr22a]
4) *Glamsham* [Gla22]
5) *LyricsOff* [Lyr22c]
6) *LyricsMINT* [Lyr22b]

Even though the tool proved useful, there were two setbacks. First was the need for the usage of *Google API keys* [Goo22] that have a limited daily use. Second, was the fact that some songs could still not be found. About 100 songs fit into this category. In order to present the best possible results, a manual effort was made to fetch the lyrics. This was toilsome and time consuming effort, being a perfect example of the idea of diminishing returns. Despite the scrupulous process, there are 18 tracks with no lyrics to be found. This was a slightly surprising finding, but explainable given the fact that these tracks belonged to less known and older artists. We decided not to invest any further effort into these. For some of them, it would have been viable to listen to the song and write down the lyrics, but it would be a process with a very low repeatability and difficult to replicate. This process of adding missing lyrics was done on top of the previous steps and doesn't change the structure of the data nor it's functionality. As such, it is an optional extra step that can be removed from the pipeline without compromising its successful execution.

B.2.2) Remarks

The fact that amongst the aggregate set of what are regarded as *500 best albums ever released* and the *The 50 Greatest Latin Albums of the Past 50 Years*, there are songs whose lyrics were not found on the internet rises questions about the criteria used to give those albums their classification. Even though an album quality is not dependent on the availability of its lyrics, it was to be expected that someone would have, even if independently and with no official backing, have uploaded said lyrics to the internet.

C. Information retrieval improvement

Along the Section III, several retrieval systems were tested. These variations included different schemas and different configurations for each query. For this milestone, the query configurations previously experienced were replaced by a single one, for all the queries. However, two different schemas were compared. Given the introduction of several languages support, the schemas had to suffer slight alterations. In order to properly evaluate the modifications made, the information needs were refined and can be found in A.3.

C.1) Single query configuration

To take all the Information Needs into consideration, the ”edismax” parser was used to encompass the queries previously made with either ”lucene” or ”dismax” parser.

The fields where to search the query were extended and boosts were given to some of them, in order to define a general relevance. The final configuration used was: *lyrics_en^4 lyrics_es^4 lyrics_other^4 artist^2 track^8 album album_release_date^0.1*. This way, it was expected to assign greater relevance to ”track” and ”lyrics” fields, and put the others in background, but allowing to search for them all.

A query phrase slop of 3 was configured to allow the search to be less restrictive.

C.2) Schemas

The schemas are the same used for and explained in M2 - Information Retrieval. The differences are when it comes to the lyrics field, which has been altered, the synonyms filter was better applied and the application order of the filters was explored. A perspective of the complete updated schema can be found in Listing A.1.

C.2.1) Simple schema

For the control group, a similar schema to ”Schema 2” presented in C.1) was used. The difference is that, to agree the language analysis, lyrics_en, lyrics_es and lyrics_other are indexed with the ”textType”(A.4)) instead of with a single field ”lyrics”, which doesn't exist anymore.

C.2.2) Updated schema

The final schema in A.4) was adapted to the new data set, generated from the lyrics' language analysis, in order to index the different lyrics fields.

The previous ”lyricsType” was converted to ”enLyricsType” and it is only used to index the ”lyrics_en” field. The synonym filter previously used was replaced by ”SynonymGraphFilterFactory” which allows to use synonyms of a given file. With this change, it was needed to find a text file containing English synonyms [8].

For Spanish lyrics, it were applied filters with a similar behavior to the ones applied in the English lyrics. It was used the filter ”SnowballPorterFilterFactory” oriented to the Spanish language in order to get the words stemmed. This filter was used instead of the ”EnglishMinimalStemFilterFactory”, ”EnglishPossessiveFilterFactory” and ”PorterStemFilterFactory” in the English lyrics type. As a text file with Spanish synonyms couldn't be found, a dictionary of Spanish words[9] was used. For each word of the dictionary, its synonyms were gathered by scrapping a Word Reference website [10].

C.2.3) Filters application order

After a deeper exploration of the filters to be used, a doubt about its application order emerged. The stemming and synonyms processes seem to be incompatible. Applying a stem filter before a synonym filter can make some words not findable because the synonym file doesn't have the words stemmed. In other way, applying the synonyms filter before the stem filter isn't either very efficient, since some words with different verb forms would be find in synonyms file for example.

Therefore, one can conclude that there is no perfect way of ordering these two filters. However, it was chosen the first

[8] https://github.com/zaibacu/thesaurus/blob/master/en$_{t}hesaurus.jsonl$

[9] https://github.com/titoBouzout/Dictionaries/blob/master/Spanish.dic

[10] https://www.wordreference.com/sinonimos/

one because it seems to be the one, out of the two, which can have better results.

D. Evaluation

D.1) Manual evaluation process

Manual evaluation followed the same procedure as was described in Section Manual evaluation process, only results were limited to 20. All the metrics were calculated based on the relevant documents retrieved that can be found in A.4.

D.2) System comparison

Metrics on both systems were calculated to help identify which performed better. No comparisons with the schemas of M2 - Information Retrieval are made as the queries used were different and, as such, not comparable.

D.2.1) Average Precision

The average precision for the queries in both schemas can be seen in the Table III.

Query	Simple schema	Updated schema
1	0.6804	0.8696
2	0.5631	0.6120
3	0.7250	0.5322
4	0.6956	1.000
5	0.9140	1.000
6	0.4762	0.4500
7	0.7595	0.9519

Table III: Average precision by query for each system

D.2.2) P@10

The P@10 for the queries in both schemas can be seen in the Table IV.

Query	Simple schema	Updated schema
1	0.6	0.8
2	0.5	0.5
3	0.6	0.3
4	0.6	0.1
5	0.8	0.8
6	0.3	0.2
7	0.6	1.0

Table IV: P@10 by query for each system

D.2.3) Mean Average Precision

The average precision of the produced schemas can be seen in the Table V.

Simple schema	Updated schema
0.6877	0.7737

Table V: P@10 by query for each system

D.2.4) P-R curves

For some of the queries, the difference in performance is not very significant (Fig. 21, Fig. 22.

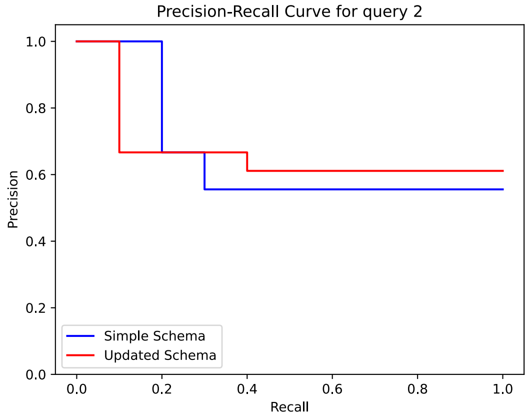

Fig. 21: Query 2 P-R Curve: Milestone 3

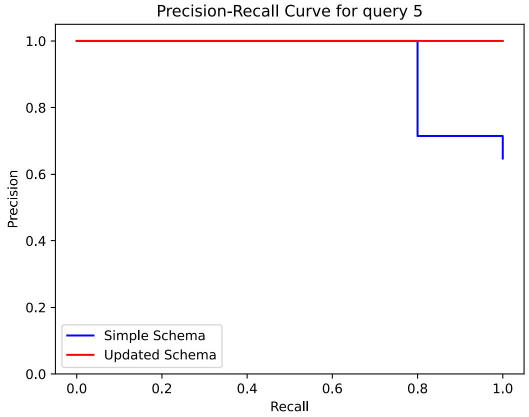

Fig. 22: Query 5 P-R Curve: Milestone 3

For two queries, the simple schema yields a clearly better result. (Fig. 23, Fig. 24).

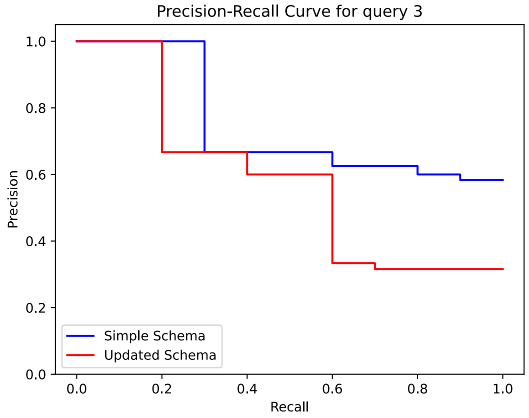

Fig. 23: Query 3 P-R Curve: Milestone 3

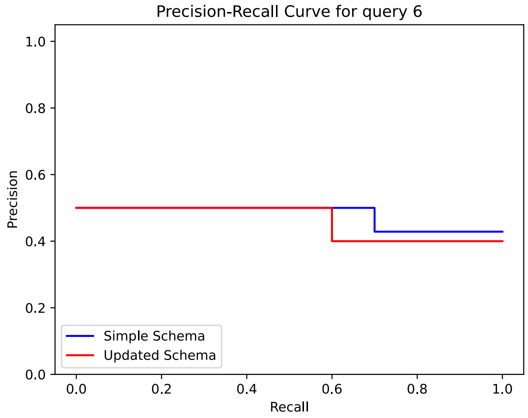

Fig. 24: Query 6 P-R Curve: Milestone 3

For the rest of the queries, the new schema provided the better results (Fig. 25, Fig. 26, Fig. 24, Fig. 27).

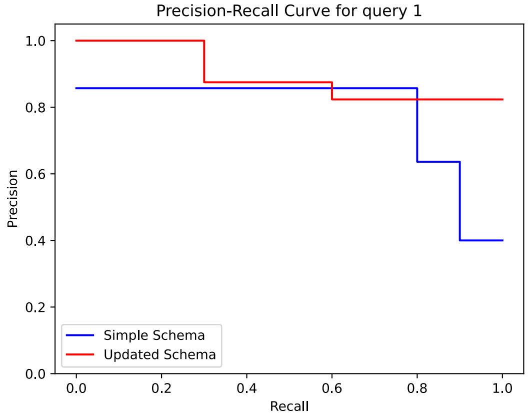

Fig. 25: Query 1 P-R Curve: Milestone 3

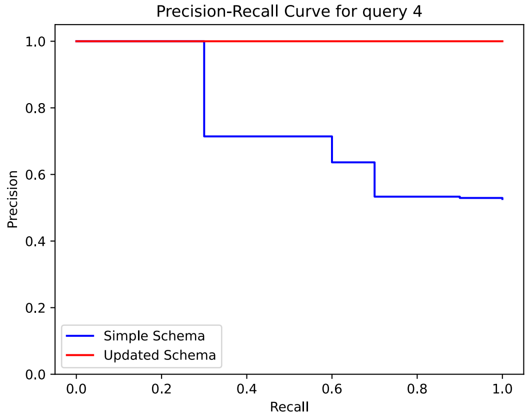

Fig. 26: Query 4 P-R Curve: Milestone 3

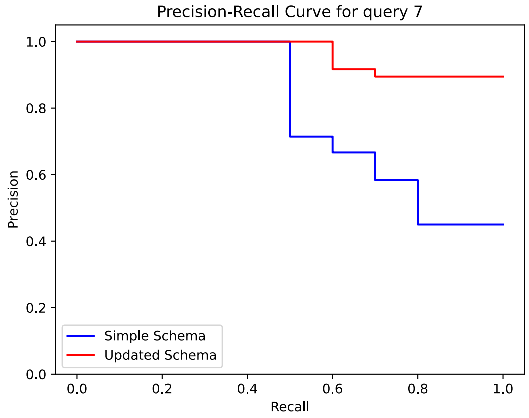

Fig. 27: Query 7 P-R Curve: Milestone 3

D.2.5) Comparison

Looking at the precision metrics and at the PR-curves, we can see that the new schema had an overall better performance. In fact, 11 percentage points of improvement, when it comes to the Mean Average Precision, is very significant.

Despite that, we can still point out some flaws with the schema, mainly when it comes to the synonyms. Due to an oversight, the synonyms are being applied after the stemming, rendering many of them useless because a stemmed word simply won't match with any of the synonyms. Further more, the synonyms files were not looked into in detail and feature many equivalences with a low correlation that may lead to non-relevant documents being chosen with no apparent reason. An example of this is the letter ”I” being associated with the chemical element iodine, and associated concepts, and to a set of numbers. A narrower synonyms list would be a good starting point for new improvements in the system.

V. Conclusion

A vast set of music was analysed in this project, gathering almost 8,000 songs covering a time period of more than 80 years. During the development process, it has become evident that the process of creating an information system is complex. Firstly, gathering a data set from the very inception is not a trivial task. There are a lot of factors that determine the quality of the final data. Since the project relies on gathering data from outsides sources, as opposed to collecting points of data ourselves, the reliance on external entities makes the whole process more prone to inconsistencies as was the case. The pipeline designed aims to take into account all of the susceptibilities of the composing parts and work around those. It can't, however, account for unavailable data. Secondly, the translation of an information need to a query is hard and only in more advanced systems is this done reliably. Finally, the evaluation of the system is done manually, which is inefficient, but the only reliable way to determine the relevant documents of a query. Improvement were made to take into account multiple languages and sources of lyrics. The main objectives were accomplished with the building and evaluation of the search system.

References

[Gen09] Genius. *About Genius*. 2009. URL: https://genius.com/Genius-about-genius-annotated (visited on 10/12/2022).

[Sta11] Billboard Staff. *About Billboard Magazine*. 2011. URL: https://www.billboard.com/music/music-news/about-billboard-magazine-467736/ (visited on 12/14/2022).

[Lam14] Paul Lamere. *Welcome to Spotipy!* 2014. URL: https://spotipy.readthedocs.io/en/master/# (visited on 10/12/2022).

[Sta15] Billboard Staff. *The 50 Greatest Latin Albums of the Past 50 Years*. 2015. URL: https://www.billboard.com/photos/50-most-essential-latin-albums-past-50-years-6686047/ (visited on 12/14/2022).

[L19] Jake L. *How Long Should a Song Be? (5 Things to Consider)*. 2019. URL: https://musicianport.com/how-long-should-a-song-be/ (visited on 11/16/2022).

[Mag20] Rolling Stone Magazine. *The 500 Greatest Albums of All Time*. 2020. URL: https://www.rollingstone.com/music/music-lists/best-albums-of-all-time-1062063/ (visited on 10/12/2022).

[Wik20] Wikipedia. *Wikipedia:WikiProject Albums/500*. 2020. URL: https://en.wikipedia.org/wiki/Wikipedia:WikiProject_Albums/500 (visited on 10/12/2022).

[Agr22] Rishabh Agrawal. *lyrics-extractor*. 2022. URL: https://pypi.org/project/lyrics-extractor/ (visited on 12/14/2022).

[Gen22a] Genius. *Genius Home Page*. 2022. URL: https://genius.com (visited on 10/12/2022).

[Gen22b] Genius. *Genius TERMS OF SERVICE*. 2022. URL: https://genius.com/static/terms (visited on 11/16/2022).

[Gen22c] Genius. *The Genius verified artist*. 2022. URL: https://genius.com/verified-artists (visited on 10/12/2022).

[Gla22] Glamsham. *Glamsham*. 2022. URL: https://www.glamsham.com/ (visited on 12/14/2022).

[Goo22] Google. *Google Custom Search JSON API*. 2022. URL: https://developers.google.com/custom-search/v1/overview (visited on 12/14/2022).

[Lyr22a] LyricsBell. *LyricsBell*. 2022. URL: https://www.lyricsbell.com/ (visited on 12/14/2022).

[Lyr22b] LyricsMINT. *LyricsMINT*. 2022. URL: https://lyricsmint.com/ (visited on 12/14/2022).

[Lyr22c] LyricsOff. *LyricsOff*. 2022. URL: https://www.lyricsoff.com/ (visited on 12/14/2022).

[Lyr22d] Lyricsted. *Lyricsted*. 2022. URL: https://lyricsted.com/ (visited on 12/14/2022).

[Sea22] Elastic Search. *Elastic Search Home Page*. 2022. URL: https://www.elastic.co/pt/ (visited on 11/16/2022).

[Sol22] Solr. *Solr Home Page*. 2022. URL: https://solr.apache.org (visited on 11/16/2022).

[Spa22] Spacy. *Spacy*. 2022. URL: https://spacy.io/ (visited on 12/14/2022).

[Spo22a] Spotify. *About Spotify*. 2022. URL: https://newsroom.spotify.com/company-info/ (visited on 10/12/2022).

[Spo22b] Spotify. *Spotify Terms of Use*. 2022. URL: https://www.spotify.com/us/legal/end-user-agreement/ (visited on 11/16/2022).

[Wik22] Wikipedia. *Wikipedia:About*. 2022. URL: https://en.wikipedia.org/wiki/Wikipedia:About (visited on 10/12/2022).

APPENDIX

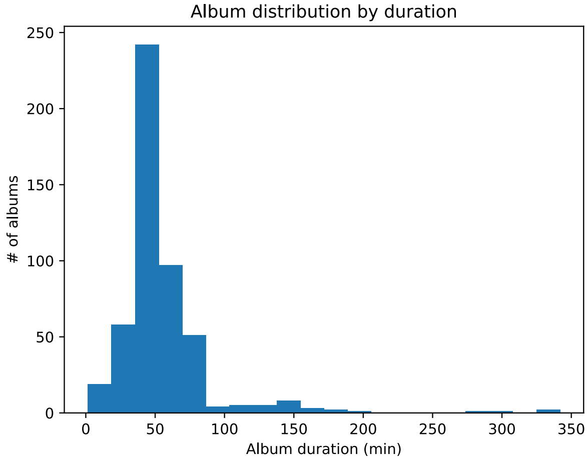

Fig. A.1: Album distribution by duration

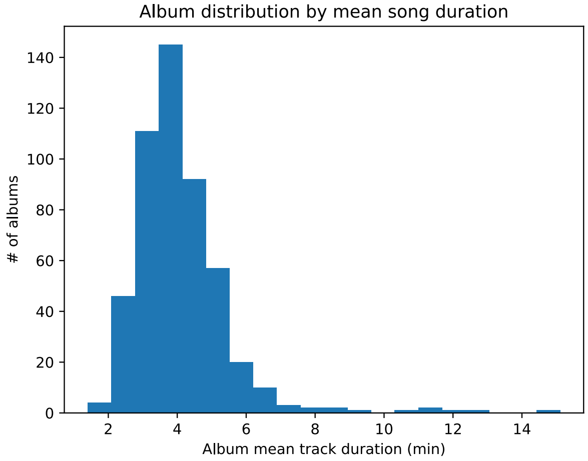

Fig. A.2: Album distribution by song duration

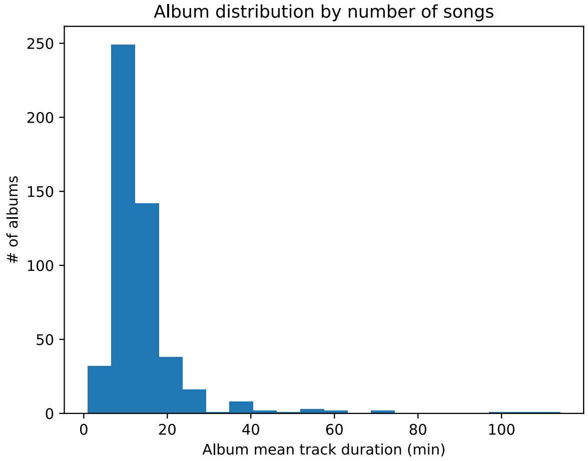

Fig. A.3: Album distribution by number of songs

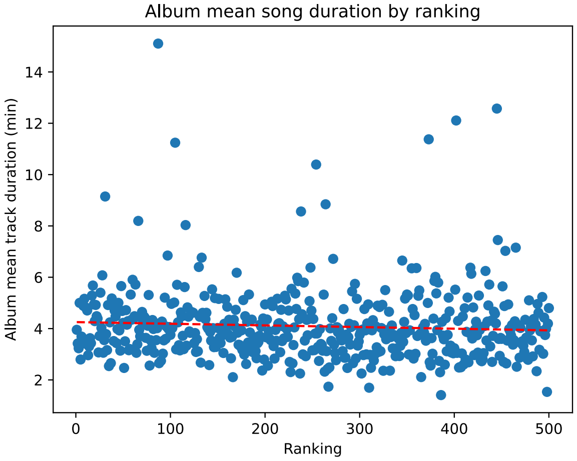

Fig. A.4: Album mean song duration by ranking

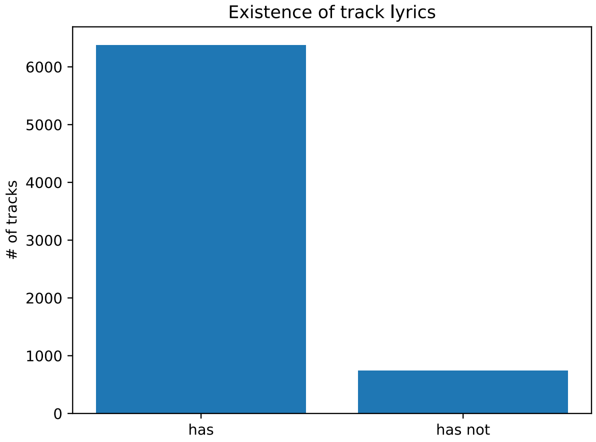

Fig. A.5: Lyrics existence

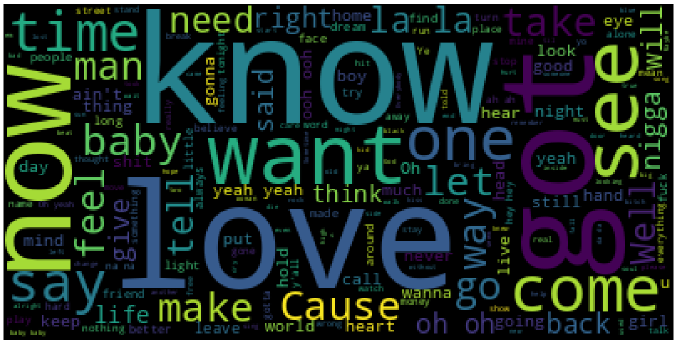

Fig. A.6: Wordcloud

ID	Information Need	Query
1	I want 1991 Nirvana songs having between 2 and 3 minutes	defType: dismax q: nirvana 1991 qf: artist album_release_date fq: !frange l=120 u=180track_duration rows: 30
2	I want songs with a regretting tone	defType: dismax q: regret2̂ sorrow pain qf: track2̂ lyrics rows: 30
3	I want an underrated relaxing song	defType: dismax q: calm enjoy peace quiet qf: track album artist album_realease_date lyrics bf: field(album_ranking) rows: 30
4	I want a song that speaks of love in a depressing way	defType: dismax q: love5̂ -good bad -happy sad qf: lyrics rows: 30
5	I want a song that talks of life and love	defType: lucene q: lyrics: ”life love” 3 q.op: AND rows: 30
6	I want songs about surprise and happiness	defType: lucene q: track: surpr* track: happ* rows: 30
7	I want a song with a sentence like ”I like her”	defType: dismax q: ”I like her” qf: tracks lyrics qs: 5 rows: 30
8	I want the very best songs	defType: dismax q: 1 rank qf: album_ranking rows: 30

Table A.1: Information needs to query for Milestone 2

SxQy	Documents																													
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
SFinalQ1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
SFinalQ2	1	1	0	0	1	1	0	0	0	0	0	1	1	1	0	1	0	0	0	0	0	1	0	0	1	0	0	0	0	0
SFinalQ3	1	1	0	1	0	1	1	0	1	0	1	0	0	0	1	1	0	0	0	1	1	0	0	0	0	1	0	1	0	0
SFinalQ4	0	0	1	1	1	1	1	1	1	1	0	1	0	1	1	0	1	0	0	1	0	0	0	1	0	1	0	0	0	1
SFinalQ5	1	1	1	1	1	1	1	0	1	0	1	1	0	1	1	1	0	1	0	0	1	1	1	1	1	0	0	0	0	1
SFinalQ6	0	1	1	1	1	0	1	1	0	1	0	0	1	0	1	0	1	1	0	1	1	1	1	0	1	1	0	1	1	1
SFinalQ7	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
SFinalQ8	1	1	1	1	1	1	1	1	1																					
S1Q1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
S1Q2	0	0	0	1	1	1	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
S1Q3	1	1	1	0	0	1	0	1	1	1	0	0	0	1	0	0	1	1	0	0	0	0	1	1	0	0	0	0	0	0
S1Q4	1	1	1	1	1	1	1	1	0	1	0	0	0	1	1	0	1	1	0	0	0	0	0	0	1	0	1	1	0	0
S1Q5	1	1	1	1	1	0	1	1	1	1	1	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
S1Q6																														
S1Q7	1	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
S1Q8	1	1	1	1	1	1	1	1	1																					
S2Q1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
S2Q2	1	1	0	0	1	1	0	0	0	0	0	1	1	1	0	1	0	0	0	0	0	1	0	0	1	0	0	0	0	0
S2Q3	1	1	1	0	0	1	0	1	1	1	0	0	0	1	0	0	0	1	1	0	0	0	0	1	1	0	0	0	0	0
S2Q4	0	0	1	1	1	1	1	1	1	1	0	1	0	1	1	0	1	1	0	0	0	0	0	1	0	1	0	1	0	0
S2Q5	1	1	1	1	1	1	0	1	0	1	1	1	1	1	0	1	0	0	1	1	1	1	1	0	0	0	0	0	0	0
S2Q6	0	1	1	1	1	0	1	1	0	1	0	0	1	0	1	0	1	1	0	1	1	1	1	0	1	1	0	1	1	1
S2Q7	1	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
S2Q8	1	1	1	1	1	1	1	1	1																					

Table A.2: Relevant documents obtained in III-C

ID	Information Need	defType: edismax qf: *lyrics_enˆ4 lyrics_esˆ4 lyrics_otherˆ4 artistˆ2 trackˆ8 album album_release_dateˆ0.1* qs: 5 rows: 20
1	I want songs with a regretting tone	q: regretˆ2 sorrow pain
2	I want an underrated relaxing song	q: calm enjoy peace quiet
3	I want a song that speaks of love in a depressing way	q: loveˆ5 -good bad -happy sad
4	I want a song that talks of life and love	q: lyrics_en: ”life love” 3
5	I want songs about surprise and happiness	q: surprise happiness
6	I want a song with a sentence like ”I like her”	q: ”I like her”
7	I want a Spanish song that talks about love and life	q: amor vida

Table A.3: Information needs to query for Milestone 3

SxQy	Documents																			
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
SSimpleQ1	0	1	1	1	1	1	1	0	0	0	1	0	0	0	0	0	0	0	0	1
SSimpleQ2	1	0	1	0	0	0	1	1	1	0	0	0	0	1	1	1	1	1	0	0
SSimpleQ3	1	1	0	0	1	1	0	1	0	1	0	1	0	0	0	0	0	0	0	0
SSimpleQ4	1	1	0	1	0	1	1	0	0	1	1	0	0	0	1	0	1	0	1	0
SSimpleQ5	1	1	1	1	1	1	1	1	0	0	0	0	1	1	0	0	1	0	0	0
SSimpleQ6	0	1	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
SSimpleQ7	1	1	1	1	0	0	1	0	1	0	0	1	0	0	0	0	0	0	1	1
SUpdatedQ1	1	1	1	1	0	1	1	1	0	1	1	0	1	1	1	1	1	0	0	0
SUpdatedQ2	1	0	0	1	1	1	0	0	1	0	0	1	1	1	1	0	1	1	0	0
SUpdatedQ3	1	0	1	0	1	0	0	0	0	0	0	1	0	0	0	0	0	1	1	0
SUpdatedQ4	1																			
SUpdatedQ5	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0
SUpdatedQ6	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
SUpdatedQ7	1	1	1	1	1	1	1	1	1	1	0	1	0	1	1	1	1	1	1	0

Table A.4: Relevant documents obtained in IV-D

```
Field types
    - textType
        - tokenizer (index and query analyzer)
            - StandardTokenizerFactory
        - filters (index and query analyzer)
            - ASCIIFoldingFilterFactory
            - LowerCaseFilterFactory
    - enLyricsType
        - tokenizer (index and query analyzer)
            - StandardTokenizerFactory
        - filters (index analyzer)
            - ASCIIFoldingFilterFactory
            - LowerCaseFilterFactory
            - EnglishMinimalStemFilterFactory
            - EnglishPossessiveFilterFactory
            - PorterStemFilterFactory
            - RemoveDuplicatesTokenFilterFactory
        - filters (query analyzer)
            - ASCIIFoldingFilterFactory
            - LowerCaseFilterFactory
            - EnglishMinimalStemFilterFactory
            - EnglishPossessiveFilterFactory
            - PorterStemFilterFactory
            - RemoveDuplicatesTokenFilterFactory
            - SynonymGraphFilterFactory
    - esLyricsType
        - tokenizer (index and query analyzer)
            - StandardTokenizerFactory
        - filters (index analyzer)
            - ASCIIFoldingFilterFactory
            - LowerCaseFilterFactory
            - SnowballPorterFilterFactory
        - filters (query analyzer)
            - ASCIIFoldingFilterFactory
            - LowerCaseFilterFactory
            - SnowballPorterFilterFactory
            - RemoveDuplicatesTokenFilterFactory
            - SynonymGraphFilterFactory

Fields
    - artist: textType
    - album: textType
    - album_release_date: pint
    - album_ranking: pint
    - n_tracks: pint
    - track: textType
    - track_duration: pint
    - lyrics_en: enLyricsType
    - lyrics_es: esLyricsType
    - lyrics_other: textType
```

Listing A.1: Updated schema