Billboard 200

João Miguel, José Azevedo, Ricardo Ferreira
Information Description, Storage and Retrieval, DAPI
Master in Informatics and Computing Engineering, MIEIC
Faculty of Engineering, University of Porto, FEUP
{up201604241,up201506448,up200305418}@fe.up.pt

ABSTRACT

Billboard 200 is a record chart which ranks the top 200 music albums on a weekly basis. This chart is published by the Billboard magazine in the United States for more than 30 years. The charts are frequently used to convey the popularity of artists. This data was crossed with information on artists, musical genres, lyrics and more, allowing the creation of new datasets with rich content about the most popular music since 1963 to 2019. These datasets have been divided into four types: Albums, Artists, Tracks and Ranks. They were analysed, conceptually described and made ready for being incorporated on an Information Retrieval tool. During the analysis it was possible to find some patterns about the changes on the music industry like for example the increased number of songs per album or the increased diversity of artists in the Billboard 200 charts. While evaluating the Information Retrieval tool, Apache Solr proves to be a good choice, providing powerful filters and analyzers and also highly customizable to fit the collections in analysis. The final information system provides very high precision, with values near 98%, an increased of 30% regarding values without the usage of custom filters and analyzers. To finalize the analisys of this dataset, concepts of web semantic have been applied and created an ontology, the Billboard 200 Ontology. Available ontologies, like the Music ontology, have been analysed and served has basis to the creation of the Billboard 200 Ontology. This ontology lead to some changes in the conceptual model, for example to better identify classes that have tags or to create better relations between authors and their work. This ontology was then queried with SPARQL which proved to be a very powerful tool to infer about the dataset. In the end, a comparison between information retrieval and web semantics is done as well as a reflexion on possible applications.

1 INTRODUCTION

The *Billboard 200* is a popular platform that publishes a weekly chart for the top 200 most popular albums on that week. This chart is the music industry reference for popularity of albums in the United States and is active since 1945 [20]. The format of the chart changed through the years, it started with only five positions and published without a clear frequency, updates were between three to seven weeks. In 1956 the Billboard chart started to be published weekly with the 10 to 30 best selling albums and in 1963 it started to publish 150 positions, it continues to change until, in 1967, publishing 200 positions which are the positions published still today. These changes were driven by the changes in the music industry, in the formats the albums were released and in the increase and diversity of artists and musical genres that appeared throughout the years. Using the *Billboard 200* [1] chart as a base, this work tries to create an information access system about music. The goal is to have in one place and easily accessible information on albums, musics and artists since 1963 to 2019. The platform will provide ranks on the *Billboard 200*, albums names, release dates, artists, bands, biographies, song lyrics, characterization by musical genre and performing queries to get and order this information. This system will act has an access point for the history of music, at least of the most popular music, between 1963 and 2019.

In this document, it is described the preparation and characterization of the proposed dataset and the use of an information retrieval tool on the proposed dataset and their exploitation with free-text queries. Concepts of Web Semantics are also applied and an ontololy created to be later queried with SPARQL.

Related to the preparation and characterization of the dataset, it is described which datasets are used for this project and its sources (Sec. 2), how was the data collected, cleaned and enriched (Sec. 3), a conceptual model for the datasets (Sec. 4), the return documents and possible search tasks of the platform (Sec. 5) and the data characterization of the data collected (Sec. 6).

Related with the use of the information retrieval tool and exploitation of queries, it is described the tool used, comparing it to other available tools (Sec. 7), the documents analysis and indexable components (Sec. 8) and the indexing and retrieval processes, including weighting experimentation and results evaluation (Sec. 9).

Related with web semantics and the ontology created, it starts with a study on existing ontologies (Sec. 10), then it is provided a description of the ontology developed (Sec. 11), query exploration of the ontology with *SPARQL* and results analysis (Sec. 12), evaluation of the developing tool used (Sec. 13), a comparison between web semantics and information retrieval tools (Sec. 14) and possible applications for the ontology developed (Sec. 15).

2 DATA SOURCES

Three sources of information are used for this project. One structured dataset with the *Billboard 200* charts which comes in a SQL database. The other sources come from two websites, Metro Lyrics and Last.Fm and provide unstructured data. With this information new datasets will be built. The datasets used for this project are presented in details in the next subsections.

2.1 *Billboard 200* Datasets

The main dataset used is a database provided by the *Components One* group [11], a publication and research group that assembles, investigate and editorializes large datasets with various members from different backgrounds and located around the United States of America. This database is free for use [12] and covers ranking charts from 1963 to the first month of 2019. Two tables from this dataset are used, the *Albums* table and the *Acoustics* table.

The *Albums* table provides information about the albums that were nominated for the *Billboard 200* ranking, more specifically, the date it was introduced on the rank, the name, the artist, as well as the length of the album.

The *Acoustics* table provides information about the songs of the albums selected for the *Billboard 200*, which include the name of the song, the album, the artist, as well as a number of technical features about the songs, like the acousticness, danceability, duration, energy, instrumentalness, key, liveness, loudness, mode, speechiness, tempo, time_signature and valence. But this extra information will not be used since we consider that it doesn't add valuable information.

In total, this dataset covers 33,011 albums, 9,675 artists and 339,854 songs.

2.2 Last.fm

Last.fm [14] is a music website founded in the United Kingdom in 2002 [21]. This website provides a handful of information about music, which is used to complement the albums, songs and artists, that can be used for personal and non-commercial purposes [13]. The process we used to extract and treat this information is described in the Section 3.2. The data obtained is stored on *JavaScript Object Notation* (*JSON*) files.

2.3 MetroLyrics

MetroLyrics [4] is a lyrics-dedicated website, founded in December 2002[22]. It is used to obtain the lyrics of the songs from the albums on *Billboard 200.* The data obtained is stored in *JavaScript Object Notation* (*JSON*) files and can be used for personal and non-commercial purposes [3]. More about the process of extraction can be found on Section 3.2.

3 DATA PREPARATION

The main dataset used is very organized and needs minimal action to extract the relevant data. Besides, since it is stored in a *SQLite* database, it is also easy to query. In the same way, the information retrieved via web scrapping is also very organized and easy to get and process. The pipeline used to extract and enrich the data is outlined in the Figure 1.

This pipeline has two main purposes, to clean the data and enrich it. This two operations will be described in more detailed in the next sections.

3.1 Data Cleaning

The first stage of the pipeline from Figure 1 is the data cleaning, accomplished through the use of the tool *OpenRefine* [6]. This tool is used to remove empty entries from the database and to remove entries with extra characters. This operation will result in two files:

- Albums - A file with all the albums and their ranks in the *Billboard 200* since 1963 to 2019 (published weekly);
- Tracks - A file with songs and artists that match with the albums file.

In this dataset one album can have hundreds of entries, because it can be featured in the *Billboard 200* for several weeks or even months or years, this imposes a problem in the next phase of the pipeline, the scraping data. This will lead to an album to be processed several times and the pipeline becomes extremely inefficient and slow. To overcome this problem, and since we cannot discard any entry (we would end up losing the rank information) from the previous files, the pipeline generates an auxiliary dataset with duplicates removed, without discarding the original dataset. This is done in the step 1 and 2 marked in the pipeline. In these steps all the characters are also escaped to HTML format to be used to generate the URL addresses for *MetroLyrics* and *Last.Fm.* Both websites have simple and intuitive URLs using the name of the album, the name of the music or the name of the artist. For example to get information about the band Pink Floyd in Last.FM the URL is "https://www.last.fm/music/Pink+Floyd" and to get the lyrics for the music Wish you were here in MetroLyrics the URL is "https://www.metrolyrics.com/wish-you-were-here-lyrics-pink-floyd.html".

3.2 Data Enrichment

The second stage of the pipeline is the data enrichment. To enrich the datasets already obtained from *Billboard 200*, the websites *Last.fm* and *MetroLyrics* were crawled using the *Scrapy* framework [17].

The crawler uses the cleaned files stated in the previous section and load these files into 4 dataframes using *Pandas* [15], an open source data analysis and manipulation library for *Python*:

- **ranks:** Table loaded from albums.csv with information of the albums positions on the *Billboard 200* charts on different dates.
- **albums:** Table with albums information obtained from albums.csv by separating the albums columns and removing duplicates.
- **tracks:** Table loaded from tracks.csv.
- **artists:** Table with artists information obtained from tracks.csv by separating the artists columns and removing duplicates.

Both *Last.fm* and *Metrolyrics*, like already said, have user-friendly links, than can be built following a common structure and using the information from the columns of the dataframes above:

LASTFM_URL = https://www.last.fm/music ML_URL = https://www.metrolyrics.com

- LASTFM_URL/**{artist}**/**{subset}** : Links to pages with information of an artist.;
- LASTFM_URL/**{artist}**/_/**{song}**/**{subset}** : Links to pages with information of a song;
- LASTFM_URL/**{artist}**/**{album}**/**{subset}** : Links to pages with information of an album;
- ML_URL/**{song}**-lyrics-**{artist}**.html : Links to pages with the lyrics of a song.

The **{artist}**, **{song}** and **{album}** fields can be obtained from the dataframes to search for specific albums, artists and tracks. The **{subset}** section indicates what kind of information we want to obtain for that album, artist or song, which can be either "tags", "wiki", or even empty if we want an overview.

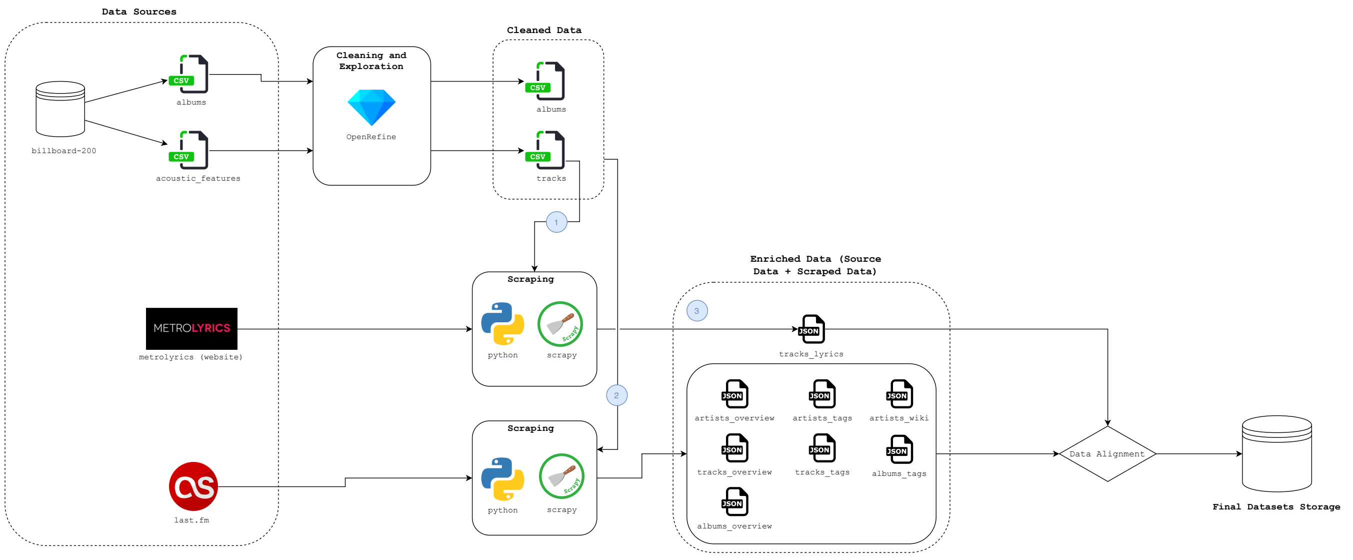


Figure 1: Data Pipeline.

Each spider searches for a subsect of all artists, albums or tracks and exports that information to a *JSON* file. As a result, the crawler obtains 8 *JSON* files with scraped data:

- **albums_overview:** Number of listeners and release date of the albums.
- **albums_tags:** List of tags for each album.
- **tracks_lyrics:** Lyrics of the tracks.
- **tracks_overview:** Number of listeners and duration of the tracks.
- **tracks_tags:** List of tags for each track.
- **artists_overview:** Number of listeners of the artists.
- **artists_tags:** List of tags for each artist.
- **artists_wiki:** The biography and number of listeners of all artists. If the artist is an individual (Solo) it also contains its birth date and birth location. If the artist is a group of individuals (Band) it also contains the location of the foundation, years of activity and a list of its members.

The next step is to complement the dataframes above with the scraped data that was stored in the *JSON* files. Each *JSON* file has the keys to the albums, artists or tracks which the scraped data corresponds to, so the information can be aggregated. This aggregation would result in 4 main tables: **ranks**, **albums**, **tracks** and **artists** (solo and band).

4 DATA MODEL

Figure 2 shows the final data model. Which is comprised of the following elements:

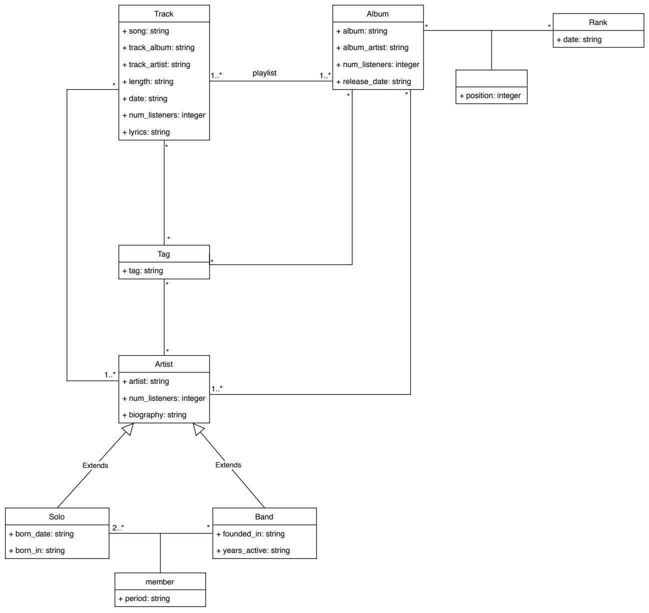


Figure 2: Conceptual Data Model.

- **Rank:** List of top albums on the *Billboard 200* chart on a specific date. An album can be at the top of the chart on different dates, so it can be associated to several ranks and in different positions between them.
- **Album:** Saves all the information of an album, such as name of the album, name of the artist, release date, number of tracks, total duration and number of listeners.
- **Artist:** Saves general information of an Artist, such as name of the artist, number of listeners and its biography (textual information). An Artist can also be one of two subclasses

that hold more specific information, depending on whether it is a Solo (referring to an individual) or a Band (group of individuals).
- **Solo:** Saves specific information of an individual artist, such as birth date and birth location.
- **Band:** Saves information of a group of artists that perform together, such as foundation location, years of activity, and the list of members (can be Solo artists, if their information is provided by the datasets).
- **Tag:** A tag contains a string that can label many albums, artists or songs, giving important information about them, such as genre.

5 INFORMATION RETRIEVAL TASKS

The information described in the previous sections will feed a platform that will allow searches and will return relevant documents based on the search parameters.

5.1 Returned Documents

Each search will return one or more documents. There are 4 types of documents for this project:

- **Albums**: It provides album information as well as its songs and artist;
- **Tracks**: It provides song information as well as its lyrics;
- **Artists**: It provides artist information as well as its albums;
- **Ranks**: It provides an album ranking chart corresponding to a date.

5.2 Possible Search Tasks

Several search queries are possible and can be simple or very complex. For example, one user can be interested in all information about an artist and type in the query only the artist name, this users expects to see results has songs from this artists, albums, ranks and of course the artist document. Other users could be interested in one album and type in the query field the name of the album and probably also the artist name, expecting to retrieve the album document, its songs and the artist document.

The possible search tasks and the possible documents returns for this project are:

- **Rank by date (year, month, day)**: returns albums, artists and ranks;
- **Artists (band or solo)**: returns artists and albums;
- **Location**: returns artists;
- **Album**: returns album, artist, songs and best rank;
- **Release date (year, month, day)**: returns albums and artists;
- **Musical genre**: returns albums, artists and songs;
- **Songs (by name or words/sentences from lyrics)**: returns songs.

6 DATA CHARACTERIZATION

The dataset in analysis is very big, having 574,000 entries. Given the nature of the data, the same album can have hundreds of entries in the database, because really popular entries will be featured in the *Billboard 200* for several years, or months. Since this dataset is in a *SQLite* database we can characterize the data with relative ease. In this dataset the album with the most entries is *The Dark Side of the moon* with 942 entries. Another particularity which increases the complexity of the analysis is the quantity of albums named *Greatest Hits* from different artists, this represents 5,905 entries. To overcome this, the scripts always consider the pair album and artist.

The number of albums per year is constant from one year to the other has can be seen in Appendix A - Figure 5, which was predictable, being the variations due to repeating albums in multiple weeks. The year 2019 has only data featuring the month of January. Looking at the number of songs per year, Appendix A - Figure 6, we can see that the number of songs included in the albums has increased over the years, being the year 2014 the year with most songs in the *Billboard 200*. This increase throughout the years happens because of the increasing number of songs per album. Similar conclusions can be taken relatively to the number of artists with albums in the rank, Appendix A - Figure 7, this can be a consequence of the increase number of diffusion mediums, which put more artists on the spotlight. Also, we study the average length of songs lyrics per year, Appendix A - Figure 8, observing the song length tended to increase throughout the 60s to the late 90s, decreased a bit through the late 2000s and early 10s and increasing again towards the latter years from the dataset.

7 INFORMATION RETRIEVAL TOOLS EVALUATION

To create a suitable information retrieval tool, Elastic Search [9] and Apache Solr [10] have been put side by side and compared. This analysis does not focus on performance, because the data collection is not so big, all the documents together have less than 500MB, and both tools are prepared to index and query terabytes of data. Both tools are build on top of Apache Lucene, supporting similar features but they differ in terms of query language, deployment and other functionalities.

Apache Solr provides search capabilities through HTTP requests with powerful features such as distributed full-text search, faceting, near real time indexing, high availability, NoSQL features, among others and uses the Lucene Syntax for queries which is very robust and allows very complex queries.

Elasticsearch uses a RESTful API's to provide its index and search functionalities and it archives the distribution of data on multiple servers using the shards concept, providing horizontal scaling, distributed full-text search, a powerful query language, multitenancy and others. Elasticsearch is completelly based on JSON and is suitable for time series and NoSQL data.

Apache Solr has native request handlers to ingest data from various sources, XML files, CSV files, databases, JSON and more with a simple post command. Elasticsearch on the other hand uses the Beats [8] tool to ingest formats other than JSON. These two tools support custom analyzers, synonym-based indexing, stemming and various tokenization options which is a desirable feature for the work under analysis. Both tools are open source and have very active communities which is a good point because it is easy to get help in the community.

From the above, both tools could be used to index and query the datasets under analysis but in the end, Apache Solr is more appropriate, it is lightweight than Elasticsearch, easier to set up and requires less resources to run which is desirable for this small project. Apache Solr is also more suitable for static data, like the one under analysis, while Elasticsearch is more focused on scaling and data analytics, features that will not be used. For all this, the next sections will use Apache Solr to query the data.

8 DOCUMENTS AND INDEXABLE COMPONENTS

Four types of documents could be returned after each query, like described on Section 5. Each of these documents has different fields with different importance for the search that will be exploited to tune the search and compare different approaches.

The *Artist* document contains the field **artist** with the name of the artist, **num_listeners** with the number of listeners, **tags** with a list of tags related with the artist, **born_date** the birth date of the artist, **born_in** the born location of an artist, usually the city and country, **died** date of the artist death if applicable, **years_active** with the number of years a band is active, not applicable to solo artists, **founded_in** location of foundation of a band, usually the city and country and the field **members** with a list of the names of the band members. In this document **born_date**, **born_in**, **died**, **years_active**, **founded_in** and **members** can be null if not applicable.

The *Album* document contains the field **album** with the name of the album, **album_artist** with the name of the artist, **num_listeners** with the number of listeners, **release_date** with the date the album was released, **tags** with a list of tags related with the album and **playlist** with a list of the songs included in the album.

The *Track* document contains the field **song** with the name of the song, **track_artists** with the name of the artist, **track_album** with the name of the album were the music is included, **date** the release date of the song, **length** the length of the song in minutes, **num_listeners** the number of listeners and **lyrics** with the lyrics for the song.

The *Rank* document contains the field **date** with the date of the rank list and the field **ranks** with a list of albums, their artist and position on the rank.

All four documents could be related with each other by the artist name and some documents have even more relations as specified in the Data Model, Figure 2. The index processes will focus on textual fields because they allow the usage of more filters and analyzers and because all the fields that relate the documents with each other are textual. Fields like dates or numbers will be left to be indexed by the default analyzers already present in Apache Solr.

8.1 Schemas, filters and analyzers

The collections will be indexed using two different schemas, one using the default filters and analyzers already present on Apache Solr and the other using custom filters and analyzers to allow smarter and improved search results and queries. In the default mode Solr uses the mode "field guessing" feature, where Solr attempts to guess what type of data is in a field while it's indexing it. It is also called the "schemaless" mode.

The default filters schema uses the **text_general** field type which apply simple filters like tokenization, stop words and lower case. The custom approach uses three custom field types: **artist-name**, **tag-text** and **descriptive-text**.

The **artist-name** applied to fields that hold the artists names uses the following analyzers:

- **PatternReplaceFilter**: Used to replace the characters *$* and *!* by *s* and *i*, respectively, since several artists use those special characters to replace those letters: artists such as *P!nk*, *A$ap Rock*, *$uicideBoy$* should also be identified as *Pink*, *Asap Rocky* and *SuicideBoys*. Then, it is used to remove the characters . and /, so artists such as *N.W.A* and *AC/DC* can also be identified by *NWA* and *ACDC*.
- **LowerCaseFilter**: Converts text to lower case, so the artists names are case insensitive.
- **ManagedSynonymGraph**: Various artists and bands may be known by different names. For example, the band known as *Red Hot Chili Peppers*, is also quite often referred to by its acronym *RHCP*. This filter allows to register those similarities. (searching by *rhcp* will return the same results as searching for *Red Hot Chili Peppers*).

The **tag-text** field applied to tags, uses the following analyzer:

- **PatternReplaceFilter**: Removes the *s* character from the end of tags that refer to years or decades, such as *80s* and *90s*. Thus, these tags are also identified when the user searches without the *s* (*80* and *80s* can be used interchangeably).

The **descriptive-text**, applied to fields that hold extensive and descriptive text, such as biography and lyrics, derives from the standard *text_general* and adds the following analyzers:

- **StopFilter**: Removes stop words
- **EnglishPossessive, EnglishMinimal and PorterStemFilter**: Used to stemming in order to obtain the base form of the words of a text. For example, words such as *win*, *winned* and *winning* will match each other.

9 INDEXING AND RETRIEVAL PROCESSES

To test the adequacy of the filters, analyzers and indexing strategies exposed in the previous section, three different information needs will be used. Each query explores a different behaviour of the indexing strategy built. Besides this queries, it will also be put together a set of weights for each of the relevant fields to change the way the results are ordered and influencing the relevancy of the top results. The first 20 results will be compared.

9.1 Information needs and queries

TOPIC#1

Title: The album M.A.A.D city by Kendrick Lamar

Description: Find out information about the album M.A.A.D City and its artist.

Narrative: Relevant results will be the document of the album itself, documents of songs included in the album and documents of the ranks where the album was present.

Query: `kendrick lamar maad city`

TOPIC#2

Title: The artist P!nk

Description: Get information about the artist P!nk.

Narrative: Searching for the work "pink" it is very likely that a user is looking for the artist P!nk, it's albums, musics and ranks. So relavant documents here would be documents related with the artists and not with the word pink.

Query: `pink`

TOPIC#3

Title: Artists born or bands active during the 80s decade that won or have been nominated for a Grammy award.

Description: Find artists (solo or bands) that have born or have been active in the 80s and won or have been nominated for the Grammies.

Narrative: Relevant documents would be artists from the 80s, born or active in that decade, that have references to winning or nomination to the Grammy awards in the description.

Query: `(born_date:198? OR years_active:198?)`
`AND biography:"Grammies"`

9.2 Weights

The Table 1 shows the weights that were applied in the retrieval process. The weights are applied to the most relevant fields, the ones that have more information and the ones that are thought to have the information that a user is seeking. The weights have been tuned to provide three special behaviours. If the query only has the name of the artist, the first result is the artist document, followed by albums from that artists and then songs. If the query has references to the album, the first document returned is the album, followed by the songs in that album followed by documents about the artists and ranks. Finally, if the query is about a song the first document returned will be the song followed by albums that have that song in their playlist.

Document Field	Weight
artist	3.2
album_artist	2.8
track_artist	2.6
album	2.4
track_album	2.2
rank.album	2.0
song	1.8
playlist	1.6

Table 1: Weights applied per relevant fields.

9.3 Evaluation

The query from TOPIC#1 uses the name of the album and the artist and shows very interesting results with and without custom analyzers and with and without filters. For start, this query gives 10,690 results without the use of custom analyzers and only 5,114 results with analyzers, showing that the use of analyzers narrows the results, making the search more accurate and less comprehensive, for example, results that only have one of the therms of the search query are ignored. The use of weights did not change the number of results but improves the ordering of the results, giving much more relevant results at the top. In Table 3, included in Appendix B it is possible to see which were the relevant documents in the first 20 results for the case without filters or analyzers and for the case with analyzers and filters and the improve on quality from one case to the other is very relevant. Without filters non relevant documents show up on the 7th result and using filters and weights the non relevant document only shows up at the 19th position. Also, the number of relevant and non relevant documents is very different on the two cases, using custom analyzers and weights there's only 2 non relevant results and without there are 11 non relevant results. Tracing a Precision-Recall graph (Figure 10) it is possible to see that the use of custom analyzers and weights improves significantly the query results showing precision of 1.0 for all recall values against a reduce of precision for values of recall above 0.6. The influence of weights is also very evident when looking at the first result which is the album document and without weights the first result is a song document. This behaviour shows that the weights and analyzers used are very appropriate for this query and fulfill the objectives pointed earlier. For TOPIC#2 the influence of the analyzers is even more evident. Without custom analyzers all the documents that include the word "pink" are returned, but none of the results is related with the artist "P!nk" which makes the top 20 results non relevant for this information need (Appendix B, Table 4). The **PatternReplaceFilter** changes completely this search returning results with the pattern "P!nk" as expected, passing from results with zero precision to results with precision higher than 50% in almost all the 20 first results. The last information need, TOPIC#3 uses a query with specific fields were two of them does not use custom analyzers, born_date and years_active and one that use the analyzer *descriptive-text*, the **biography**. The query without the use of custom analyzers returns only 1 value, and with the analyzer returns 155 results. This difference is easy to understand because the query uses the therm "Grammies" and only one result in the entire collection refers to the Grammy Award in the plural form. With custom filters and analyzers the plural and singular forms are merged in the results and the quality and quantity of the results increases like can be seen on Appendix B in the TOPIC#3 tables and graphs. Overall the use of custom analyzers and filters improves significantly. The Minimum Average Precision increases more than 30%, from 64,7% to 97,6% and the Average Precision-Recall curve for the three information needs shows significant improvement with the usage of filters, as can be seen in Figure 3 where the values of the precision are very close to 1.0 for all the recall values in the case with filters and lower than 0.75 for the case without filters.

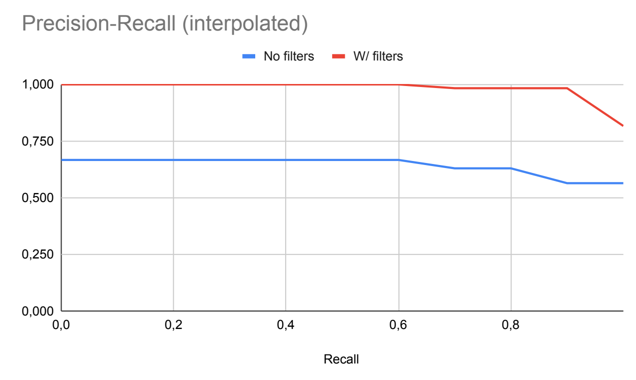


Figure 3: Average Precision-Recall curve for TOPICS 1, 2 and 3.

10 EXISTING ONTOLOGIES FOR THE DOMAIN

From a short state of the art review of existing ontologies related to music, there were two that ressembled this project subject: *The Music Ontology* [16] [5] and *Schema.org* ontology [2].

The Music Ontology, created in 2006 provides main concepts and properties to describe music in the Semantic Web, including albums and artists. It is subdivided in 3 levels:

- The first level deals with editorial information, describing superficially a song, its artists and its album;
- The second level introduces the Event ontology. This ontology can describe the workflow involving the composition of a song, and physical events in which the song was performed;
- The third level decomposes the events, going into very specific details like: in that particular moment, that person playing that instrument played that key.

Although this ontology would be more than enough to cover information about songs, artists and albums for this project, it doesn't cover information about music ranking charts, which is necessary for the *Billboard 200* chart. It is also more detailed than necessary for our use case, as the first level would be enough to cover most of the information on *Billboard 200* dataset.

The *Schema.org* ontology, founded in 2011 by Google, Microsoft, Yahoo and Yandex, is a generic ontology, which can cover multiple subjects, but not going to the level of detail encountered in more specific ontologies like *The Music Ontology* previously described. However, it seem capable of covering the dataset in study, having classes to cover tracks (*MusicComposition* and *MusicRecording*), albums (*MusicAlbum*), artists (*Person* for solo artists and *MusicGroup* for bands) and also ranks (*ItemList*), which was lacking on the previous ontology. It doesn't cover tags, but some generic fields could be adapted to fill that gap, however not the same field in all types.

With some changes both ontologies could be adapted to describe the model under study, but none of them with the level of detail required for some fields, like artists or albums tags, years active for Bands or the number of listeners. These ontologies also provide a lot of fields that the model does not need and does not have information to fill in those fields, but above all, both ontologies would require many adaptations and for those reasons a new ontology was developed and will be described in the next sections.

11 THE BILLBOARD 200 ONTOLOGY

The ontology was created using the tool *Protégé* [18]. To create the ontology the conceptual model proposed in Figure 2 was slightly changed to better represent some concepts, like individuals that have tags, work done by artists (bands and solo) and to create a better distinction between the rank and the rank positions. This new model is represented on Figure 4.

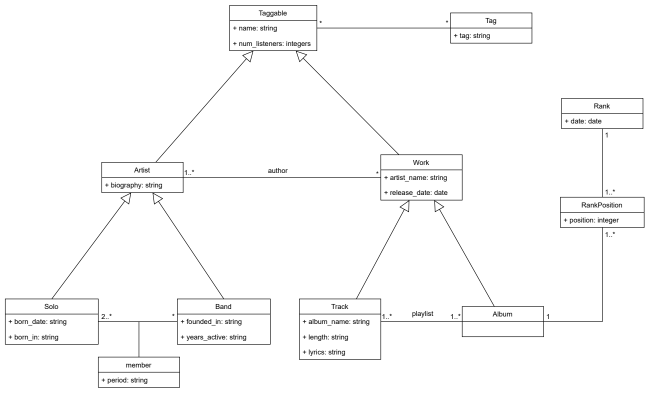


Figure 4: Conceptual Data Model updated for the Billboard 200 ontology.

In this new model all the classes that can have a tag are under the super class **Taggable**. Another significant change is the new class **Work** which is the super class of **Track** and **Album** which can be seen has work done by artists. In this new model a new class **RankPosition** was introduced to represent the position of a specific **Rank**. Each **Rank** has a specific date and is related to many individuals of **RankPosition**. Each **RankPosition** is related to only one individual of the **Rank** class and to the **Album** that stayed in that position.

11.1 Classes and Properties

The class hierarchy follows the model on Figure 4 and has the following structure:

```
owl:Thing
 ├─ Rank
 ├─ RankPosition
 ├─ Tag
 └─ Taggable
     ├─ Artist
     │   ├─ Band
     │   └─ Solo
     └─ Work
         ├─ Album
         └─ Track
```

The **Rank** holds various positions and each instance is associated with a date. The **RankPosition** represents a position of a specific **Rank** and is related to only one **Album**.

Taggable is the class of individuals that can be categorized by tags and a name and that have a number of listeners. It has two subclasses: **Artist** and **Work**.

Artist is the class to represent individuals who have created albums and/or tracks. Artists can be divided in two classes, **Band** and **Solo**. The class **Artist** has the property biography which of course is present in both its subclasses.

The class **Work**, used to represent work done by artists has two subclasses, **Album** and **Track**. Both classes have an author (**Artist**) and release date.

To relate all the classes the following Object Properties have been created:

```
owl:topObjectProperty
 ├── hasAuthor
 ├── hasInPlaylist
 ├── hasMember
 ├── hasRankedAlbum
 ├── hasRankPosition
 ├── hasTag
 ├── isAuthorOf
 ├── isInPlaylistOf
 ├── isMemberOf
 ├── isOnRank
 ├── isPositionOfRank
 ├── isTagOf
 └── wasRankedOnPosition
```

The property **hasAuthor** links an individual of the Work class (Album or Track) to its creator, who is an individual of the Artist class (Band or Solo). It is the inverse of the relation **isAuthorOf**.

hasInPlaylist links an individual of the Album class to one of its songs, which is an individual of the Track class. It is the inverse of the relation **isInPlaylistOf**.

asMember links an individual of the Band class to one of its members, which is an individual of the Solo class. It is the inverse of the relation **isMemberOf**.

hasRankPosition links an individual of the Rank to one of its positions, which is an individual of the RankPosition class. It is the inverse of the relation **isPositionOfRank**.

hasRankedAlbum links an individual of the RankPosition to the album that was in that position of the related Rank, which is an individual of the Album class. It is the inverse of the relation **wasRankedOnPosition**.

hasTag links an individual of the Taggable class (Album, Track, Band, or Solo) to one of its tags, which is an individual of the Tag class. It is the inverse of the relation **isTagOf**.

To complete the ontology the following data properties have been added:

```
owl:topDataProperty
 ├── hasActiveYears
 ├── hasBiography
 ├── hasBirthDate
 ├── hasBirthLocation
 ├── hasDeathDate
 ├── hasFoundationLocation
 ├── hasLength
 ├── hasLyrics
 ├── hasName
 ├── hasNumListeners
 ├── hasPosition
 ├── hasRankDate
 ├── hasReleaseDate
 └── hasString
```

The **hasName** and **hasNumListeners** data properties hold the name and number of listeners of an individual of the Taggable class (Band, Solo, Album, or Track).

The **hasBiography** data property holds the biography text of an individual of the Artist class (Band or Solo).

The **hasReleaseDate** data property holds the date an individual of the Work class (either Album or Track) was released.

The **hasActiveYears** and **hasFoundationLocation** data properties hold information about the years an individual of the Band class was active, and the location where it was founded.

The **hasBirthDate**, **hasBirthLocation** and **hasDeathLocation** hold the birth date and location of an individual of the Solo class, and its death date (if applicable).

The **hasLength** and **hasLyrics** hold the lenght and the lyrics of an individual of the Track class.

The **hasString** data property holds the label of an individual of the Tag class.

11.2 Restrictions

From the classes and properties described so far it is easy to understand that some restrictions must be applied to guarantee that the data is valid, coherent and respects the conceptual model. The Table 2 shows the type restrictions applied to the classes of the ontology.

A **Rank** has only one date. A **RankPosition** is associated with exactly one **Rank** by the property **isPositionOfRank**, and one **Album** by the property **hasRankedAlbum**. All individuals in the **Taggable** class have exactly one name and a single value for the number of listeners. All individuals of the class **Artist** have a biography, plus a **Band** has a period of active years and a **Solo** artist have a birth date and a birth location. All individuals in the **Work** class are associated to at least one **Artist** by the property **hasAuthor** and have a release date. An **Album** is associated to at least two individuals of the **Track** class by the property **hasInPlaylis**. All individuals of **Track** have lyrics and a length.

11.3 Data population strategy

To populate the Billboard 200 ontology, the *Cellfie Protégé* [7] plugin was used. The plugin is simple to use but has the drawback of only being compatible with data in *xlsx* format (*Microsoft Excel*). This increased the complexity to use this plugin, since all the data at the end of the data preparation pipeline (Figure 1) is in *JSON* format, divided in four different files: **albums.json**, **artists.json**, **ranks.json** and **tracks.json**.

Class	Property	Restriction
Rank	hasRankDate	exactly 1 rdfs:Literal
RankPosition	hasPosition	exactly 1 rdfs:Literal
	hasRankedAlbum	exactly 1 rdfs:Album
	isPositionOfRank	exactly 1 rdfs:Rank
Tag	isTagOf	some rdfs:Taggable
Taggable	hasName	exactly 1 rdfs:Literal
	hasNumListeners	exactly 1 rdfs:Literal
Artist	hasBiography	exactly 1 rdfs:Literal
Band	hasActiveYears	exactly 1 rdfs:Literal
	hasFoudationLocation	exactly 1 rdfs:Literal
	hasMember	min 2 rdfs:Solo
Solo	hasBirthDate	exactly 1 rdfs:Literal
	hasBirthLocation	exactly 1 rdfs:Literal
Work	hasAuthor	Some rdfs:Artist
	hasReleaseDate	exactly 1 rdfs:Literal
Album	hasInPlaylist	min 2 rdfs:Track
Track	hasLength	exactly 1 rdfs:Literal
	hasLyrics	exactly 1 rdfs:Literal

Table 2: Type restrictions applied to the ontology classes.

To adapt and convert the information, a *Python* script, using the *Pandas* library [15] was developed. This script gathers the information from the four *JSON* files and converts them to six *Excel* sheets. A sample of each of these sheets as well as the rules used to map the information with classes can be see on Appendix C. The six *Excel* sheets are *bands.xlsx*, *solos.xlsx*, *albums.xlsx*, *tracks.xlsx*, *tags.xlsx* and *ranks.xlsx* and are used to populate the analogous classes.

It is important to note that the dataset refined and enriched in the early stage of the analysis is quite extensive, as it contains data related to the Billboard ranks since 1963 and importing all of this data through the *Cellfie* plugin is very slow and resource demanding. To overcome this the script selects the last ten Billboard Ranks and the first twenty albums from each of those ranks, together with artists and tracks. With this strategy the ontology only has a sample of the data, but this sample is enough to be queried and to reasoning about the dataset like will be seen in the following sections.

12 QUERYING THE BILLBOARD 200 ONTOLOGY

A total of seven *SPARQL* queries [19] were used to test the ontology. Some are similar to the queries developed for the information needs in *Solr* and others try to explore features of *SPARQL* and how it could be used to infer new information about the dataset.

The Listing 1 shows the query used to get all the albums and their rank dates together with the artist name and rank position. The results can be seen on Figure 27 on Appendix D.

```
SELECT ?albumName ?artistName ?
    rankPosition ?rankDate
WHERE {
    ?rank a :Rank .
    ?rank :hasRankDate ?rankDate .
    ?rank :hasRankPosition ?position .
    ?position :hasPosition ?rankPosition
        .
    ?position :hasRankedAlbum ?album .
    ?album :hasName ?albumName .
    ?album :hasAuthor ?author .
    ?author :hasName ?artistName .
}
ORDER BY ?rankPosition DESC(?rankDate)
```

Listing 1: SPARQL Query #1: Get albums and their rank by date

The second query on Listing 2 gets the number of times each album has been ranked and the result can be seen on Figure 28 on Appendix D.

```
SELECT ?albumName ?artistName (count(?
    rankPosition) as ?nrOfTimesInRank)
WHERE {
    ?position a :RankPosition .
    ?position :hasPosition ?rankPosition
        .
    ?position :hasRankedAlbum ?album .
    ?album :hasName ?albumName .
    ?album :hasAuthor ?author .
    ?author :hasName ?artistName .
}
GROUP BY ?albumName ?artistName
ORDER BY DESC(?nrOfTimesInRank)
```

Listing 2: SPARQL Query #2: Get number of times each album has been ranked

The next is a simple query, just to get a list of artists and their biography. The query used is on Listing 3 and the results on Figure 29 on Appendix D.

```
SELECT ?artistName ?biography
WHERE {
    {?artist a :Solo}
    UNION {?artist a :Band}
    ?artist :hasName ?artistName .
    ?artist :hasBiography ?biography
}
```

Listing 3: SPARQL Query #3: Get artists and their biography

To get artists that have more than 800.000 listeners monthly, the query on Listing 4 was used. The results are on Appendix D on Figure 30.

```
SELECT ?artistName ?numListeners
WHERE {
    {?artist a :Solo}
    UNION {?artist a :Band}
    ?artist :hasName ?artistName .
    ?artist :hasNumListeners ?
        numListeners
    FILTER (?numListeners > 800000)
    }
ORDER BY DESC(?numListeners)
LIMIT 15
```

Listing 4: SPARQL Query #4: Get 15 artists that have more than 800.000 listeners

The query on Listing 5 gets the average number of listeners per track per album. The results can be seen on Figure 31 on Appendix D.

```
SELECT ?artistName ?albumName (AVG(?
    numListeners) as ?
    avgTrackAlbumListeners)
WHERE {
    ?track a :Track .
    ?track :hasNumListeners ?
        numListeners .
    ?track :isInPlaylistOf ?album .
    ?album :hasName ?albumName .
    ?album :hasAuthor ?artist .
    ?artist :hasName ?artistName .
}
GROUP BY ?albumName ?artistName
ORDER BY DESC(?avgTrackAlbumListeners)
```

Listing 5: SPARQL Query #5: Get track average number of listeners by album

Albums release date can be fetched with the query on Listing 6 and the results can be viewed on Figure 32 on Appendix D.

```
SELECT ?artistName ?albumName ?
    releaseDate
WHERE {
    ?album a :Album .
    ?album :hasName ?albumName .
    ?album :hasReleaseDate ?releaseDate
        .
    ?album :hasAuthor ?artist .
    ?artist :hasName ?artistName .
}
```

Listing 6: SPARQL Query #6: Get albums release date

The final query tries to understand which albums from American Artists are related with Christmas and Love. The Listing 7 shows the query used and the results are on Figure 33 on Appendix D.

```
SELECT ?artistName ?albumName ?albumTag
WHERE {
    ?album a :Album .
    ?album :hasName ?albumName .
    ?album :hasAuthor ?artist .
    ?artist :hasName ?artistName .
    ?album :hasTag ?fullTag .
    ?fullTag :hasString ?albumTag .
    ?artist :hasTag ?fullArtistTag .
    ?fullArtistTag :hasString ?artistTag
        .
    FILTER regex(str(?albumTag), "
        christmas|xmas|love")
    FILTER regex(str(?artistTag), "
        american")
}
```

Listing 7: SPARQL Query #7: Get Christmas and love albums from american artists

12.1 Result analysis

The usage of a tool that provides a powerful Query Language like *SPARQL* allows a thorough exploration of the dataset and to infer and understand the information it contains. The queries and the results presented on the previous section shows exactly this, even though not all the features of the language have been used. This is probably one of the biggest advantages when compared with the Information Retrieval tools used in this project.

Looking at the simplest queries like the Query#3 to get only the biography or Query#4 to get the number of listeners per artists or even Query#6 to get the release date of albums we can see that the results are very accurate and organized. The tabular form of the results is easy to read and can be easily configured with the *SELECT* clause of the query. This kind of information retrieval was also possible in Solr but the organization of the results in documents is much more difficult to read, since each document returns all the fields related and not only the relevant ones.

Query#5 is a little bit more complex and allows to understand which albums have more listeners per track. This metric could be used to infer on the popularity of artists and albums specially if crossed with the information on Query#2, to get the number of times each album was on ranks. With these two queries one can see if an album where all the tracks have a large number of listeners would keep this album more time on the Billboard 200 Charts. Unfortunatelly, since it was only uploaded a subset of the data those conclusions are not possible.

To filter albums by a particular characteristic, the Query#7 was used. In this query it was intended to find American Artists with Albums related with Christmas and Love. This query is in some how analogous to the information need of TOPIC#3 presented on Section 9, were it was intended to get artists from the 80's that have won a grammy, since both queries are trying to filter albums foloowing a rule. Comparing both results, the ones returned by *SPARQL* are much more accurate, also the usage of regular expressions filters can leverage the power of the query, but on the other side the query is much more complex.

The results from Query#1 shows how easily is to cross the data from different classes and still get a pretty and organized table of results. In this query, the albums, their artists and the dates and positions they were rankedd were put together. The information was ordered by date of the rank.

13 EVALUATION OF THE SEMANTIC WEB TOOLS USED

Essentially two tools have been used, *Protégé* [18] to create the ontology and *SPARQL* [19] to query the ontology.

Protégé is a free and open source editor to create ontologies. The desktop version, the one used, has an outdated interface and is in somehow confused to use since it requires a lot of clicks and to open and close several tabs to create classes, properties, restrictions and so on. There is also a cloud based version, the *WebProtégé*, with a more modern look, but with limited features. The tool served the project's purpose, since it is compliant with W3C Standards with compatibility for the usage of RDF and OWL 2. The tool also has pre installed description logic reasoners which was a good feature since it was possible to do all the tasks in the same tool. One major problem of *Protégé* was is limited capabilities to populate the ontology. Firstly because it needed external plugins, like the *Cellfie* plugin [7] and the plugins are limited in the formats supported. Secondly because the population process is very resourse demanding and such, it was only possible to upload a subset of the dataset.

Protégé comes with an editor for the *SPARQL* language and also allowed to run the queries on the ontology developed. Having this integrated with the tool makes a lot of sense, since we can create and infer over the ontology on one place with only one tool. But, again, this interface is poor and boring to use. Code highlight is not available and errors on queries are returned on the form of popups.

Regarding *SPARQL*, it is a powerfull language and its similiraties with *SQL* make its learning curve small. The language comes with ordering functionalities, very handy filters like regular expressions and other functionalities.

Overall, the tools and languages used were adequate to the context of this work.

14 COMPARISON BETWEEN WEB SEMANTICS AND INFORMATION RETRIEVAL TOOLS

Comparing the semantic web tool used *Protégé* with the information retrieval tool used *Apache Solr*, *Protégé* offered better and more complex schema implementation, allowing for stronger relations and better hierarchy between classes, making it easier to find new relationships. Although, the additional level of complexity and dept, makes it not as user-friendly as *Solr*, which was easier to use, with simple model implementation and population. While with *Solr* it was possible to populate the schema using all information on the dataset, on *Protégé* the dataset had to be cut.

Protégé also had the edge on data querying, *SPARQL* allowing for powerful queries while on *Solr* the queries were more limited, required more tuning to return the expected results, and better knowledge of the model beforehand. With *SPARQL*, it's possible to better structure query results, as you can specify which resources or properties are returned. It is also easier to group data: for example, to obtain the number of times an album was ranked on *The Billboard 200*, it is possible to obtain that information with a simple query with *SPARQL*, while on *Solr*, it is not possible with the basic query tool.

SPARQL results are also more assertive and more restricted than in *Solr*: for example, searching for the artist "P!nk" in *Solr*, the results include not only artist's information, but also albums and songs; in *SPARQL*, the query has to be much more complex to obtain the same information. However, with SPARQL, the results are always relevant, returning exactly what the query asks, while with *Solr*, not all results are relevant. Returning to the previous example, searching for the artist "P!nk" in *Solr*, returns also information about other artists (like "Pink Floyd") and songs with the word "pink".

15 POSSIBLE APPLICATIONS FOR THE BILLBOARD 200 ONTOLOGY

The developed ontology, since it was based on a popular music ranking website dataset, it's main purpose would be to describe ranking charts about music. However, it also can be used for music archives, since it is also rich in information about artists, songs and albums, without necessarily ranking them, or to complement information on other music databases or websites, like *Last.fm.*

It can also be used on some studies about music, for example, based on the biography and rank positions from their albums, try to infer characateristics and traits that popular artists tend to have, or comparing those caracteristics with other time periods, obtaining an idea about the evolution of artists throughout time.

It can also be crossed with social networks, for example, in an atempt to help new artists reach larger audiences, by giving some tips on how to interact with their fans, based on the intended target audience.

16 CONCLUSIONS

The data was characterized and its usage defined. The data chosen is very complete and will provide lots of information about music since 1963 and can act has a repository for the history of the most popular music from the period in analysis (1963 to 2019). The database with the *Billboard 200* proved to be an awesome starting point to extract information since it gives an exhaustive list of albums, artists and musics. Even though, the database has a huge list of albums it only has information about albums in the Billboard 200, lacking information about other less popular albums. This will render the final datasets incomplete, but on the other hand, since we are considering information from 1963 to 2019, all albums produced in these dates would represent datasets with giant proportions and challenges that are not in the scope of this analysis. The sources used to enrich this data also proved to be very satisfactory and with high quality information, leading to very complete groups of datasets. The process of characterization, data cleaning and enrichment performed ended in good quality documents that allowed the usage of an information retrieval tool and the return of satisfactory results.

Apache Solr, has several mechanisms to index the data, filter and analyze it, all with a simple and intuitive API. These characteristics lead to the fast creation of an information retrieval system with high precision. The returned results have high relevancy which shows that the characterization, analyzes, index and filtering applied meets the proposed requirements. Looking at the information needs presented in Section 9.1 and their results is very interesting to observe how those searches can be tunned which will greatly influence which results are returned. It can also help to understand how a search engine works and how important is to use filters with care to not manipulate the results and biased the users.

Exporting the models to Web Semantics required some changes to the conceptual model that allowed to reinforce some relations between the different classes, showing that Web Semantics is based on solid hierarchies and well defined rules and relations. Due to this, querying in this domain is more powerful and structured.

Information Retrieval has been central to the success of the Web and Web Semantics uses the same documents but enriched by machine understandable annotations. Through this article is easier to understand that Web Semantics are more adequate to the usage of artificial intelligence since finding new relationships in the data is simple and can be an automated process. Not only the queries can be more complex but also the results are more accurate and amazingly well organized. The level of complexity of the queries in Information Retrieval can also be quite high but not to the level provided by Web Semantics, but Information Retrieval has the possibility of tunning the search and completelly change the results for the same query, which in the World Wide Web is a very appreciated feature, as, for example, paid publications can be easily put on top.

REFERENCES

[1] BILLBOARD. 2020. *Billboard 200.* https://www.billboard.com/charts/billboard-200 Accessed: November 30, 2020.
[2] Schema.org Community. 2021. *Schema.org website.* https://schema.org/ Accessed: January 09, 2021.
[3] MetroLyrics Red Ventures Company. 2020. *MetroLyrics - Terms of Use.* https://redventures.com/legal/cmg-terms-of-use.html Accessed: December 1, 2020.
[4] MetroLyrics Red Ventures Company. 2020. *MetroLyrics Website.* https://www.metrolyrics.com/ Accessed: December 1, 2020.
[5] Music Ontology Contributors. 2021. *Music Ontology website.* http://musicontology.com/ Accessed: January 09, 2021.
[6] OpenRefine Contributors. 2020. *Open Refine Documentation.* https://openrefine.org Accessed: December 1, 2020.
[7] Protege Project contributors. 2018. *Cellfie Plugin Github Page.* https://github.com/protegeproject/cellfie-plugin Accessed: January 10, 2020.
[8] Elasticsearch. 2020. *Elasticsearch Beats tool.* https://www.elastic.co/beats/ Accessed: November 25, 2020.
[9] Elasticsearch. 2020. *Elasticsearch website.* https://www.elastic.co Accessed: November 25, 2020.
[10] Apache Software Foundation. 2020. *Apache Solr website.* https://lucene.apache.org/solr/ Accessed: November 25, 2020.
[11] Components One Group. 2019. *Acoustic and meta features of albums and songs on the Billboard 200.* https://components.one/datasets/billboard-200 Accessed: November 27, 2020.
[12] Components One Group. 2020. *Components One - About.* https://components.one/pages/about Accessed: December 1, 2020.
[13] CBS Interactive. 2020. *Last.fm - Terms of Use.* https://www.last.fm/legal/terms#para6 Accessed: December 1, 2020.
[14] CBS Interactive. 2020. *Last.fm website.* https://www.last.fm/ Accessed: December 1, 2020.
[15] NumFocus. 2020. *Pandas, Python Data Analysis Library.* https://pandas.pydata.org/ Accessed: December 1, 2020.
[16] Yves Raimond, Samer A Abdallah, Mark B Sandler, and Frederick Giasson. 2007. The Music Ontology.. In *ISMIR*, Vol. 2007. Citeseer, 8th.
[17] Scrapinghub and contributors. 2020. *Scrapy, A Fast and Powerful Scraping and Web Crawling Framework.* https://scrapy.org/ Accessed: December 1, 2020.
[18] Stanford University. 2020. *Protégé website.* https://protege.stanford.edu Accessed: January 10, 2020.
[19] W3C. 2013. *SPARQL Query Language for RDF.* https://www.w3.org/TR/rdf-sparql-query/ Accessed: January 10, 2020.
[20] Wikipedia. 2020. *Billboard 200 article based on Joel Whitburn books about the Billboard charts.* https://en.wikipedia.org/wiki/Billboard_200 Accessed: November 30, 2020.
[21] Wikipedia. 2020. *Last.fm on Wikipedia.* https://en.wikipedia.org/wiki/Last.fm Accessed: December 1, 2020.
[22] Wikipedia. 2020. *MetroLyrics on Wikipedia.* https://en.wikipedia.org/wiki/MetroLyrics Accessed: December 1, 2020.

A DATA CHARACTERIZATION

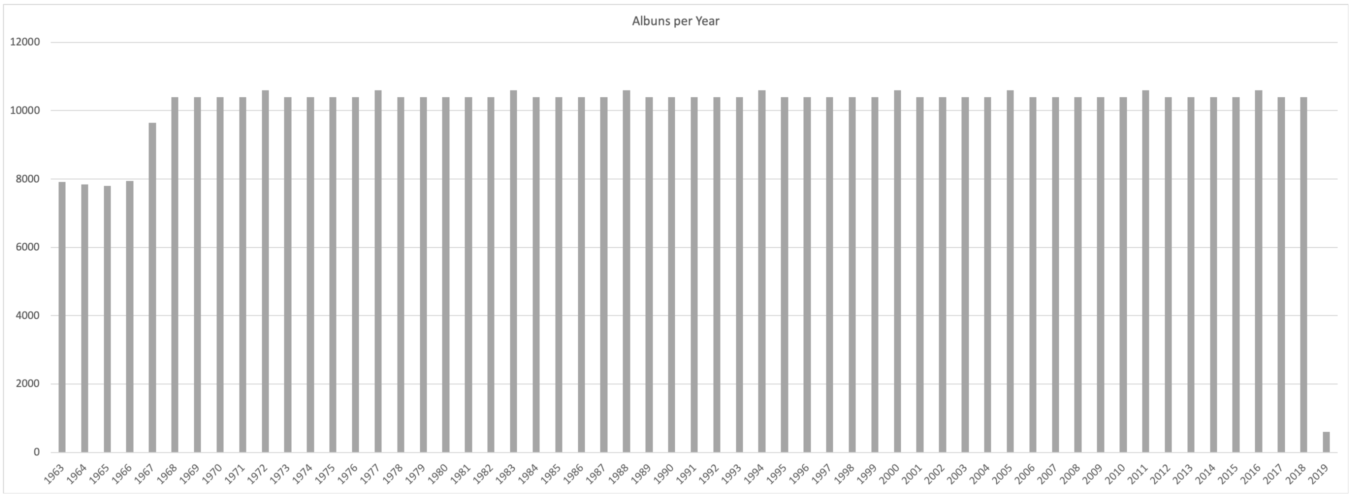


Figure 5: Albums per year in the Billboard 200.

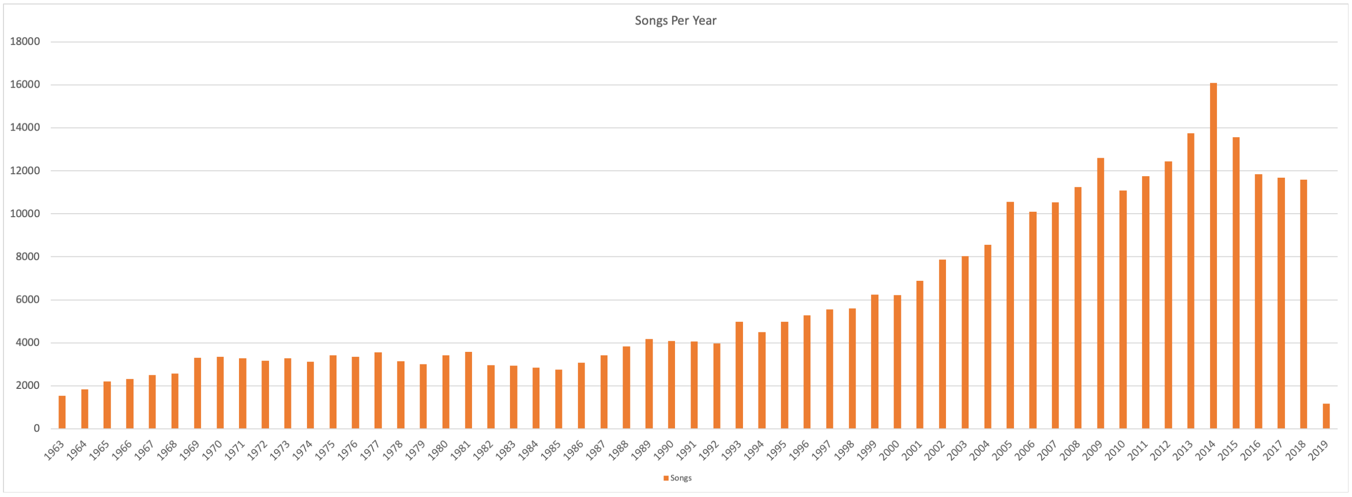


Figure 6: Songs per year in the Billboard 200.

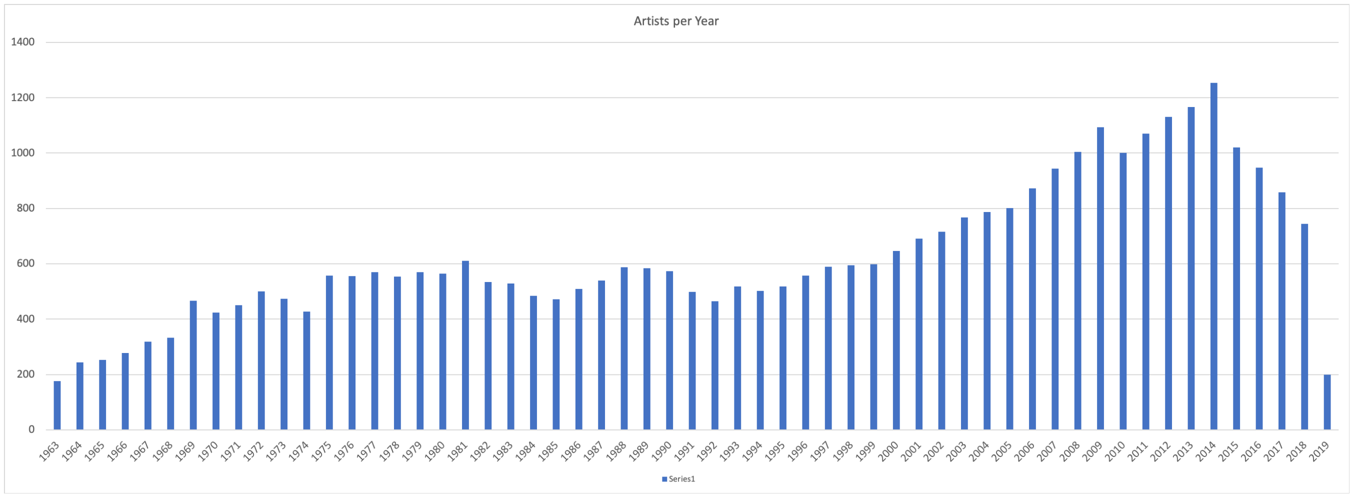


Figure 7: Artists per year in the Billboard 200.

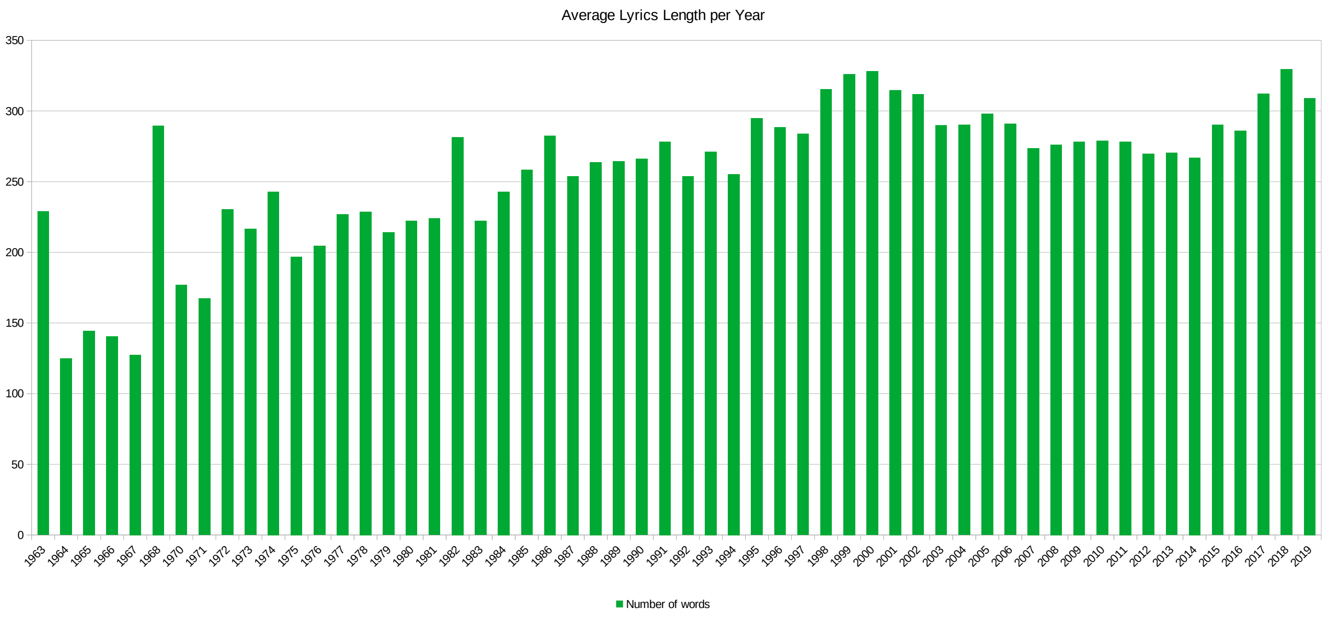


Figure 8: Average lyrics length per year in the Billboard 200.

B INFORMATION RETRIEVAL RESULTS

TOPIC#1
QUERY: `kendrick lamar maad city`

Result	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
No Filters	R	R	R	R	R	R	N	R	R	N	N	N	R	N	N	N	N	N	N	N
w/ Filters	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	N	N

Table 3: Top 20 results relevancy for TOPIC#1.

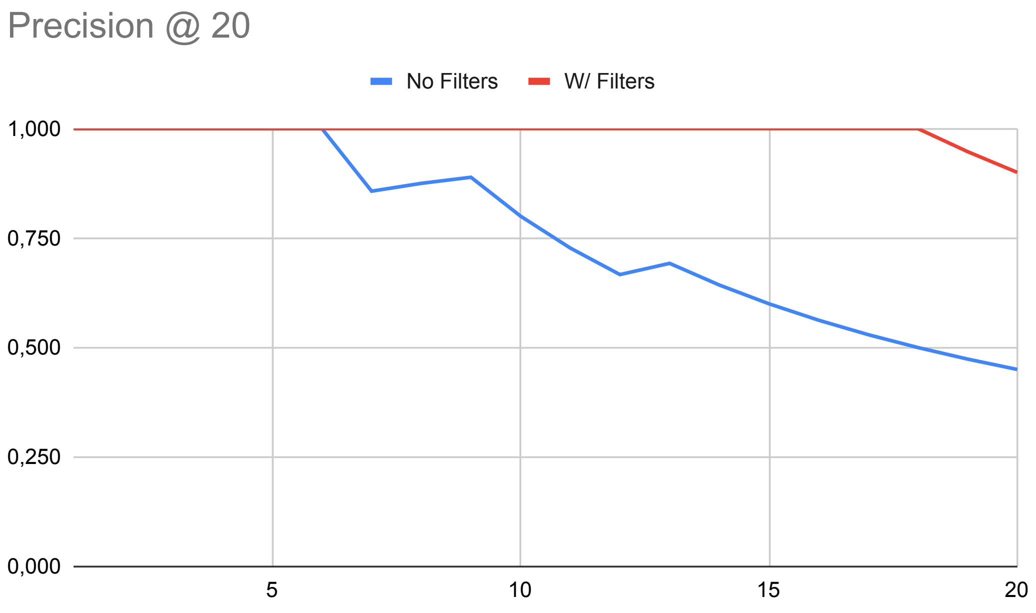


Figure 9: Precision @ 20 for TOPIC#1.

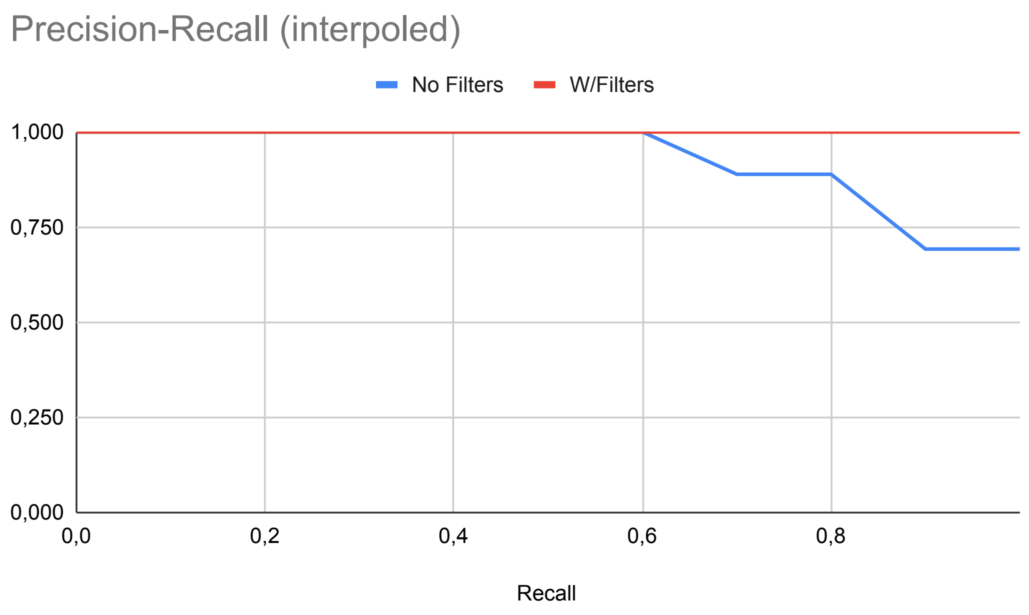


Figure 10: Precision vs Recall (interpolated) for TOPIC#1.

TOPIC#2
QUERY: `pink`

Result	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
No Filters	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
w/ Filters	R	R	R	R	R	R	R	R	R	N	N	N	N	N	N	N	N	N	N	R

Table 4: Top 20 results relevancy for TOPIC#2.

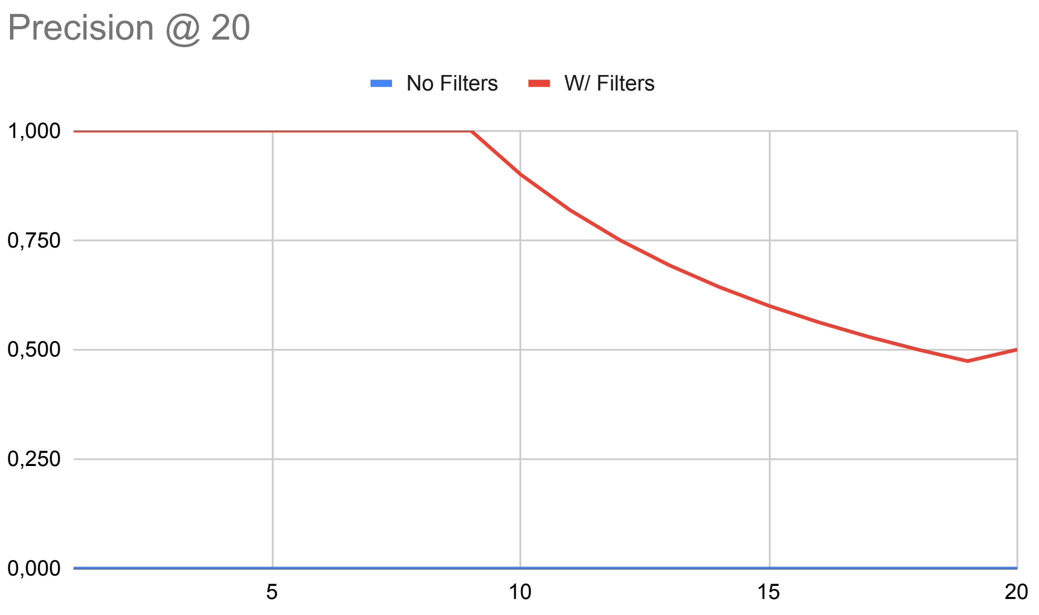


Figure 11: Precision @ 20 for TOPIC#2.

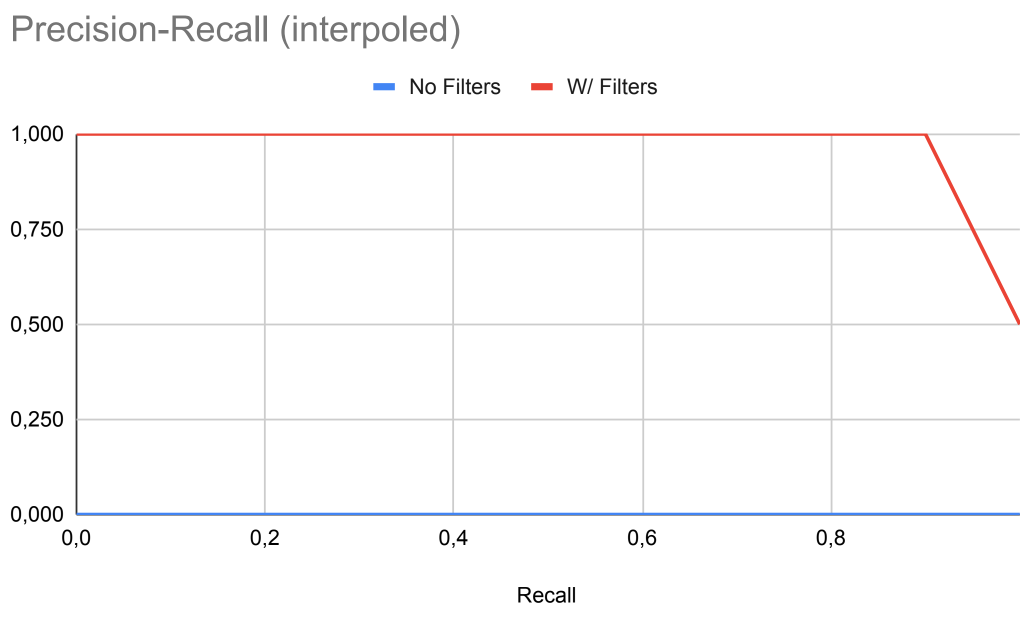


Figure 12: Precision vs Recall (interpolated) for TOPIC#2.

TOPIC#3
QUERY: `(born_date:198? OR years_active:198?) AND biography:“Grammies”`

Result	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
No Filters	R	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
w/ Filters	R	R	R	R	R	R	R	R	R	R	R	R	N	R	R	R	R	R	R	R

Table 5: Top 20 results relevancy for TOPIC#3.

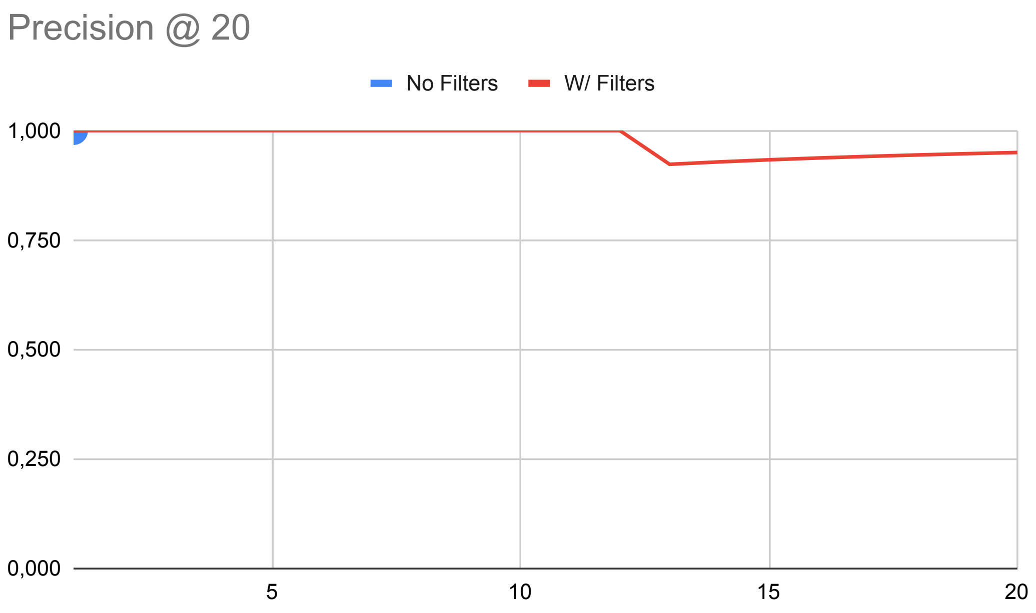


Figure 13: Precision @ 20 for TOPIC#3.

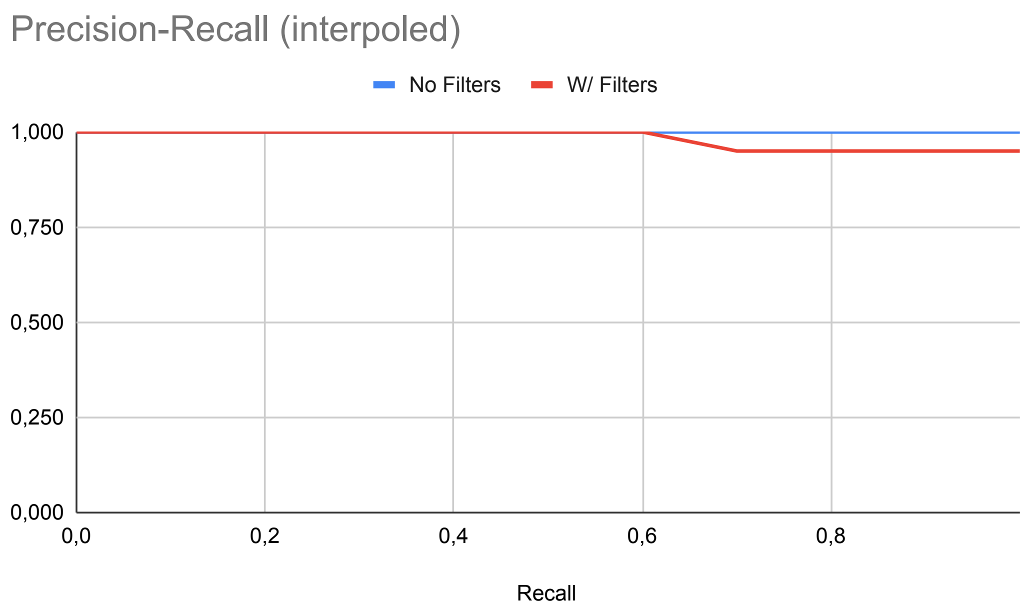
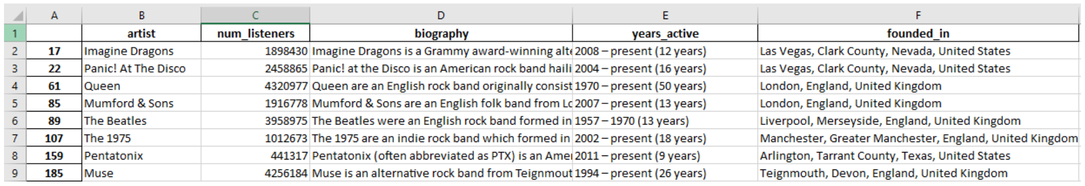


Figure 14: Precision vs Recall (interpolated) for TOPIC#3.

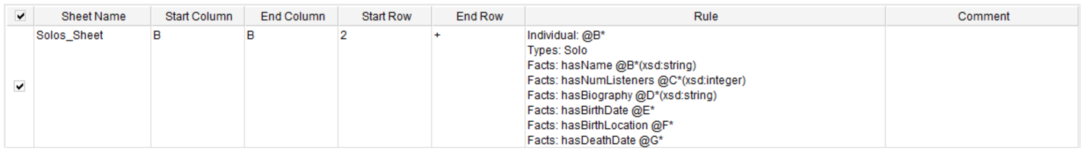
C POPULATING THE ONTOLOGY - DATA SAMPLES AND RULES

Class Band



	A	B	C	D	E	F
1		artist	num_listeners	biography	years_active	founded_in
2	17	Imagine Dragons	1898430	Imagine Dragons is a Grammy award-winning alte	2008 – present (12 years)	Las Vegas, Clark County, Nevada, United States
3	22	Panic! At The Disco	2458865	Panic! at the Disco is an American rock band haili	2004 – present (16 years)	Las Vegas, Clark County, Nevada, United States
4	61	Queen	4320977	Queen are an English rock band originally consist	1970 – present (50 years)	London, England, United Kingdom
5	85	Mumford & Sons	1916778	Mumford & Sons are an English folk band from Lc	2007 – present (13 years)	London, England, United Kingdom
6	89	The Beatles	3958975	The Beatles were an English rock band formed in	1957 – 1970 (13 years)	Liverpool, Merseyside, England, United Kingdom
7	107	The 1975	1012673	The 1975 are an indie rock band which formed in	2002 – present (18 years)	Manchester, Greater Manchester, England, United Kingdom
8	159	Pentatonix	441317	Pentatonix (often abbreviated as PTX) is an Amei	2011 – present (9 years)	Arlington, Tarrant County, Texas, United States
9	185	Muse	4256184	Muse is an alternative rock band from Teignmout	1994 – present (26 years)	Teignmouth, Devon, England, United Kingdom

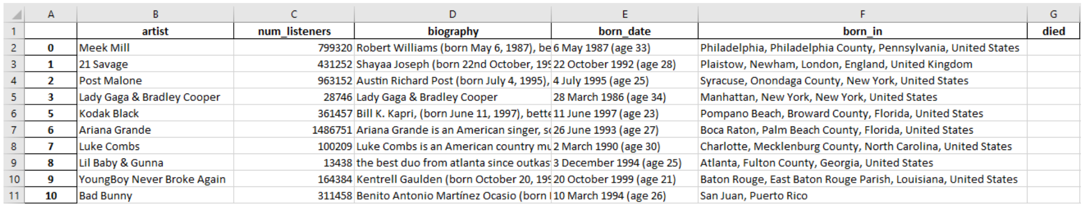
Figure 15: Sample data for the Band class in xlsx format.



✓	Sheet Name	Start Column	End Column	Start Row	End Row	Rule	Comment
✓	Solos_Sheet	B	B	2	+	Individual: @B* Types: Solo Facts: hasName @B*(xsd:string) Facts: hasNumListeners @C*(xsd:integer) Facts: hasBiography @D*(xsd:string) Facts: hasBirthDate @E* Facts: hasBirthLocation @F* Facts: hasDeathDate @G*	

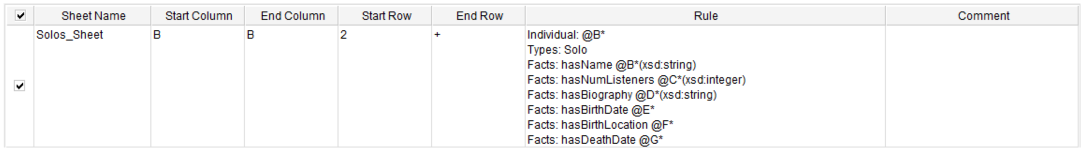
Figure 16: Rules to map the data of the Band class.

Class Solo



	A	B	C	D	E	F	G
1		artist	num_listeners	biography	born_date	born_in	died
2	0	Meek Mill	799320	Robert Williams (born May 6, 1987), be	6 May 1987 (age 33)	Philadelphia, Philadelphia County, Pennsylvania, United States	
3	1	21 Savage	431252	Shayaa Joseph (born 22nd October, 19	22 October 1992 (age 28)	Plaistow, Newham, London, England, United Kingdom	
4	2	Post Malone	963152	Austin Richard Post (born July 4, 1995),	4 July 1995 (age 25)	Syracuse, Onondaga County, New York, United States	
5	3	Lady Gaga & Bradley Cooper	28746	Lady Gaga & Bradley Cooper	28 March 1986 (age 34)	Manhattan, New York, New York, United States	
6	5	Kodak Black	361457	Bill K. Kapri, (born June 11, 1997), bette	11 June 1997 (age 23)	Pompano Beach, Broward County, Florida, United States	
7	6	Ariana Grande	1486751	Ariana Grande is an American singer, s	26 June 1993 (age 27)	Boca Raton, Palm Beach County, Florida, United States	
8	7	Luke Combs	100209	Luke Combs is an American country mu	2 March 1990 (age 30)	Charlotte, Mecklenburg County, North Carolina, United States	
9	8	Lil Baby & Gunna	13438	the best duo from atlanta since outkas	3 December 1994 (age 25)	Atlanta, Fulton County, Georgia, United States	
10	9	YoungBoy Never Broke Again	164384	Kentrell Gaulden (born October 20, 19	20 October 1999 (age 21)	Baton Rouge, East Baton Rouge Parish, Louisiana, United States	
11	10	Bad Bunny	311458	Benito Antonio Martínez Ocasio (born I	10 March 1994 (age 26)	San Juan, Puerto Rico	

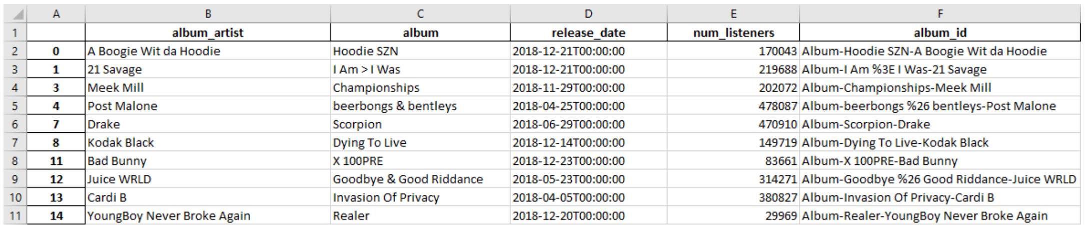
Figure 17: Sample data for the Solo class in xlsx format.



✓	Sheet Name	Start Column	End Column	Start Row	End Row	Rule	Comment
✓	Solos_Sheet	B	B	2	+	Individual: @B* Types: Solo Facts: hasName @B*(xsd:string) Facts: hasNumListeners @C*(xsd:integer) Facts: hasBiography @D*(xsd:string) Facts: hasBirthDate @E* Facts: hasBirthLocation @F* Facts: hasDeathDate @G*	

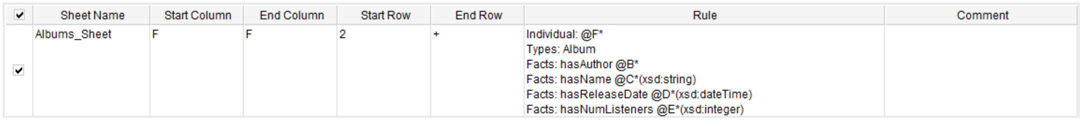
Figure 18: Rules to map the data of the Solo class.

Class Album



	A	B	C	D	E	F
1		album_artist	album	release_date	num_listeners	album_id
2	0	A Boogie Wit da Hoodie	Hoodie SZN	2018-12-21T00:00:00	170043	Album-Hoodie SZN-A Boogie Wit da Hoodie
3	1	21 Savage	I Am > I Was	2018-12-21T00:00:00	219688	Album-I Am %3E I Was-21 Savage
4	3	Meek Mill	Championships	2018-11-29T00:00:00	202072	Album-Championships-Meek Mill
5	4	Post Malone	beerbongs & bentleys	2018-04-25T00:00:00	478087	Album-beerbongs %26 bentleys-Post Malone
6	7	Drake	Scorpion	2018-06-29T00:00:00	470910	Album-Scorpion-Drake
7	8	Kodak Black	Dying To Live	2018-12-14T00:00:00	149719	Album-Dying To Live-Kodak Black
8	11	Bad Bunny	X 100PRE	2018-12-23T00:00:00	83661	Album-X 100PRE-Bad Bunny
9	12	Juice WRLD	Goodbye & Good Riddance	2018-05-23T00:00:00	314271	Album-Goodbye %26 Good Riddance-Juice WRLD
10	13	Cardi B	Invasion Of Privacy	2018-04-05T00:00:00	380827	Album-Invasion Of Privacy-Cardi B
11	14	YoungBoy Never Broke Again	Realer	2018-12-20T00:00:00	29969	Album-Realer-YoungBoy Never Broke Again

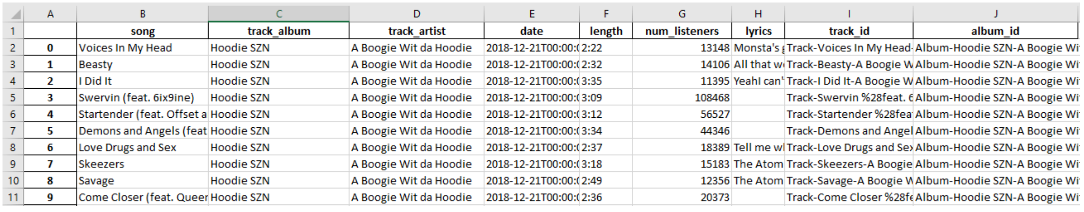
Figure 19: Sample data for the Album class in xlsx format.



✓	Sheet Name	Start Column	End Column	Start Row	End Row	Rule	Comment
✓	Albums_Sheet	F	F	2	+	Individual: @F* Types: Album Facts: hasAuthor @B* Facts: hasName @C*(xsd:string) Facts: hasReleaseDate @D*(xsd:dateTime) Facts: hasNumListeners @E*(xsd:integer)	

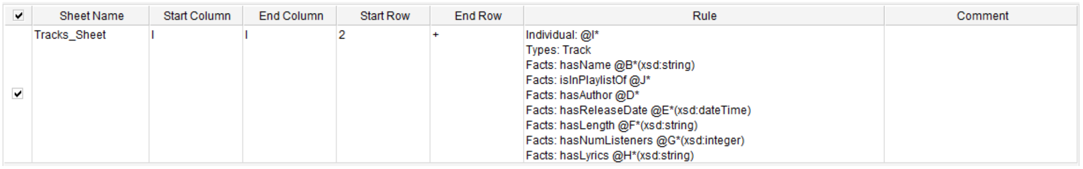
Figure 20: Rules to map the data of the Album class.

Class Track



	A	B	C	D	E	F	G	H	I	J
1		song	track_album	track_artist	date	length	num_listeners	lyrics	track_id	album_id
2	0	Voices In My Head	Hoodie SZN	A Boogie Wit da Hoodie	2018-12-21T00:00:	2:22	13148	Monsta's	Track-Voices In My Head-	Album-Hoodie SZN-A Boogie Wi
3	1	Beasty	Hoodie SZN	A Boogie Wit da Hoodie	2018-12-21T00:00:	2:32	14106	All that w	Track-Beasty-A Boogie W	Album-Hoodie SZN-A Boogie Wi
4	2	I Did It	Hoodie SZN	A Boogie Wit da Hoodie	2018-12-21T00:00:	3:35	11395	YeahI can'	Track-I Did It-A Boogie W	Album-Hoodie SZN-A Boogie Wi
5	3	Swervin (feat. 6ix9ine)	Hoodie SZN	A Boogie Wit da Hoodie	2018-12-21T00:00:	3:09	108468		Track-Swervin %28feat. 6	Album-Hoodie SZN-A Boogie Wi
6	4	Startender (feat. Offset a	Hoodie SZN	A Boogie Wit da Hoodie	2018-12-21T00:00:	3:12	56527		Track-Startender %28fea	Album-Hoodie SZN-A Boogie Wi
7	5	Demons and Angels (feat	Hoodie SZN	A Boogie Wit da Hoodie	2018-12-21T00:00:	3:34	44346		Track-Demons and Angel	Album-Hoodie SZN-A Boogie Wi
8	6	Love Drugs and Sex	Hoodie SZN	A Boogie Wit da Hoodie	2018-12-21T00:00:	2:37	18389	Tell me wl	Track-Love Drugs and Sex	Album-Hoodie SZN-A Boogie Wi
9	7	Skeezers	Hoodie SZN	A Boogie Wit da Hoodie	2018-12-21T00:00:	3:18	15183	The Atom	Track-Skeezers-A Boogie	Album-Hoodie SZN-A Boogie Wi
10	8	Savage	Hoodie SZN	A Boogie Wit da Hoodie	2018-12-21T00:00:	2:49	12356	The Atom	Track-Savage-A Boogie W	Album-Hoodie SZN-A Boogie Wi
11	9	Come Closer (feat. Queer	Hoodie SZN	A Boogie Wit da Hoodie	2018-12-21T00:00:	2:36	20373		Track-Come Closer %28fe	Album-Hoodie SZN-A Boogie Wi

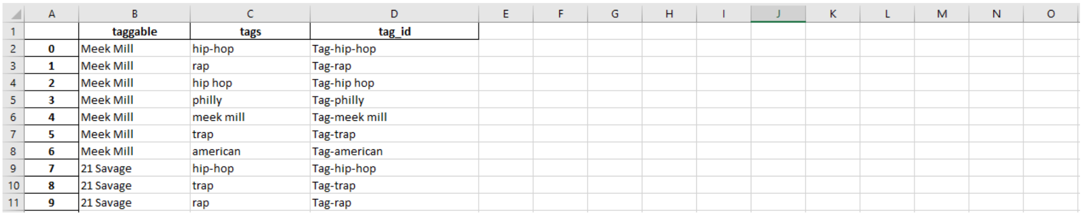
Figure 21: Sample data for the Track class in xlsx format.



✓	Sheet Name	Start Column	End Column	Start Row	End Row	Rule	Comment
✓	Tracks_Sheet	I	I	2	+	Individual: @I* Types: Track Facts: hasName @B*(xsd:string) Facts: isInPlaylistOf @J* Facts: hasAuthor @D* Facts: hasReleaseDate @E*(xsd:dateTime) Facts: hasLength @F*(xsd:string) Facts: hasNumListeners @G*(xsd:integer) Facts: hasLyrics @H*(xsd:string)	

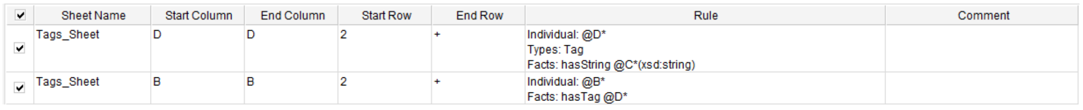
Figure 22: Rules to map the data of the Track class.

Class Tag



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1		taggable	tags	tag_id											
2	0	Meek Mill	hip-hop	Tag-hip-hop											
3	1	Meek Mill	rap	Tag-rap											
4	2	Meek Mill	hip hop	Tag-hip hop											
5	3	Meek Mill	philly	Tag-philly											
6	4	Meek Mill	meek mill	Tag-meek mill											
7	5	Meek Mill	trap	Tag-trap											
8	6	Meek Mill	american	Tag-american											
9	7	21 Savage	hip-hop	Tag-hip-hop											
10	8	21 Savage	trap	Tag-trap											
11	9	21 Savage	rap	Tag-rap											

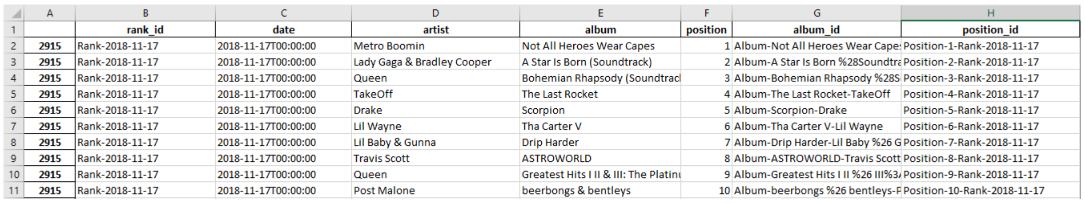
Figure 23: Sample data for the Tag class in xlsx format.



✓	Sheet Name	Start Column	End Column	Start Row	End Row	Rule	Comment
✓	Tags_Sheet	D	D	2	+	Individual: @D* Types: Tag Facts: hasString @C*(xsd:string)	
✓	Tags_Sheet	B	B	2	+	Individual: @B* Facts: hasTag @D*	

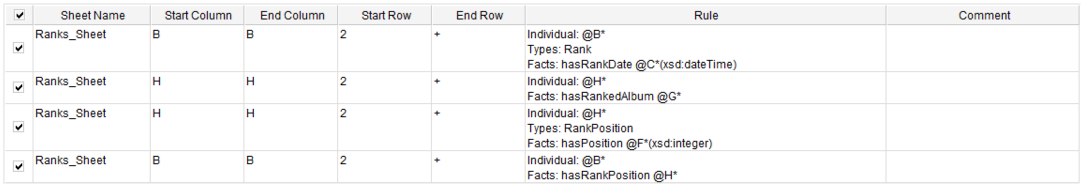
Figure 24: Rules to map the data of the Tag class.

Class Rank



	A	B	C	D	E	F	G	H
1		rank_id	date	artist	album	position	album_id	position_id
2	2915	Rank-2018-11-17	2018-11-17T00:00:00	Metro Boomin	Not All Heroes Wear Capes	1	Album-Not All Heroes Wear Cape:	Position-1-Rank-2018-11-17
3	2915	Rank-2018-11-17	2018-11-17T00:00:00	Lady Gaga & Bradley Cooper	A Star Is Born (Soundtrack)	2	Album-A Star Is Born %28Soundtra	Position-2-Rank-2018-11-17
4	2915	Rank-2018-11-17	2018-11-17T00:00:00	Queen	Bohemian Rhapsody (Soundtrac	3	Album-Bohemian Rhapsody %28S	Position-3-Rank-2018-11-17
5	2915	Rank-2018-11-17	2018-11-17T00:00:00	TakeOff	The Last Rocket	4	Album-The Last Rocket-TakeOff	Position-4-Rank-2018-11-17
6	2915	Rank-2018-11-17	2018-11-17T00:00:00	Drake	Scorpion	5	Album-Scorpion-Drake	Position-5-Rank-2018-11-17
7	2915	Rank-2018-11-17	2018-11-17T00:00:00	Lil Wayne	Tha Carter V	6	Album-Tha Carter V-Lil Wayne	Position-6-Rank-2018-11-17
8	2915	Rank-2018-11-17	2018-11-17T00:00:00	Lil Baby & Gunna	Drip Harder	7	Album-Drip Harder-Lil Baby %26 G	Position-7-Rank-2018-11-17
9	2915	Rank-2018-11-17	2018-11-17T00:00:00	Travis Scott	ASTROWORLD	8	Album-ASTROWORLD-Travis Scott	Position-8-Rank-2018-11-17
10	2915	Rank-2018-11-17	2018-11-17T00:00:00	Queen	Greatest Hits I II & III: The Platinu	9	Album-Greatest Hits I II %26 III%3	Position-9-Rank-2018-11-17
11	2915	Rank-2018-11-17	2018-11-17T00:00:00	Post Malone	beerbongs & bentleys	10	Album-beerbongs %26 bentleys-F	Position-10-Rank-2018-11-17

Figure 25: Sample data for the Rank class in xlsx format.

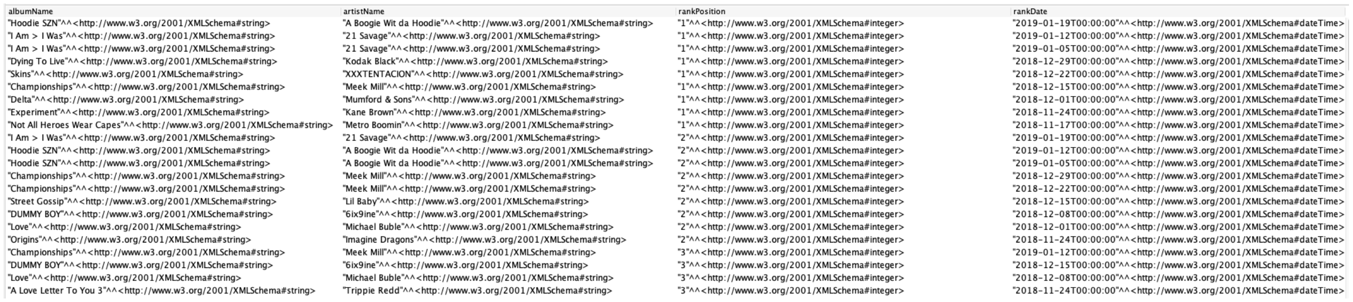


✓	Sheet Name	Start Column	End Column	Start Row	End Row	Rule	Comment
✓	Ranks_Sheet	B	B	2	+	Individual: @B* Types: Rank Facts: hasRankDate @C*(xsd:dateTime)	
✓	Ranks_Sheet	H	H	2	+	Individual: @H* Facts: hasRankedAlbum @G*	
✓	Ranks_Sheet	H	H	2	+	Individual: @H* Types: RankPosition Facts: hasPosition @F*(xsd:integer)	
✓	Ranks_Sheet	B	B	2	+	Individual: @B* Facts: hasRankPosition @H*	

Figure 26: Rules to map the data of the Rank class.

D SPARQL QUERIES RESULTS

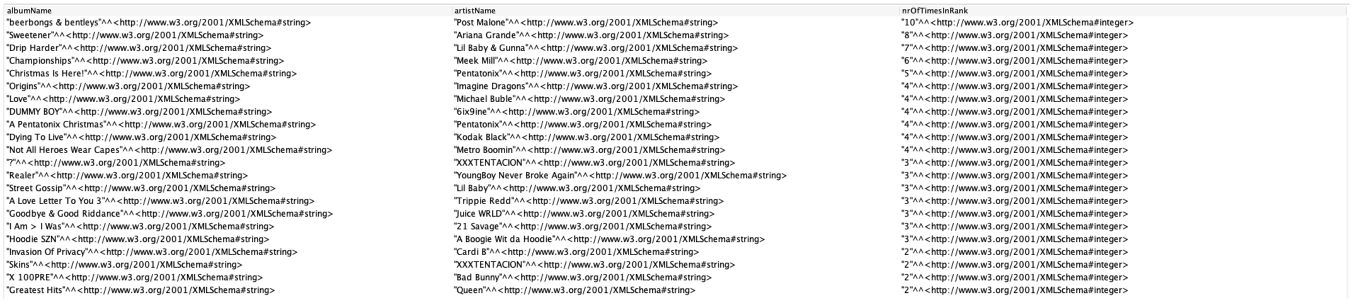
Query #1: Get Albums and their rank by date



albumName	artistName	rankPosition	rankDate
"Hoodie SZN"^^<http://www.w3.org/2001/XMLSchema#string>	"A Boogie Wit da Hoodie"^^<http://www.w3.org/2001/XMLSchema#string>	"1"^^<http://www.w3.org/2001/XMLSchema#integer>	"2019-01-19T00:00:00"^^<http://www.w3.org/2001/XMLSchema#dateTime>
"I Am > I Was"^^<http://www.w3.org/2001/XMLSchema#string>	"21 Savage"^^<http://www.w3.org/2001/XMLSchema#string>	"1"^^<http://www.w3.org/2001/XMLSchema#integer>	"2019-01-12T00:00:00"^^<http://www.w3.org/2001/XMLSchema#dateTime>
"I Am > I Was"^^<http://www.w3.org/2001/XMLSchema#string>	"21 Savage"^^<http://www.w3.org/2001/XMLSchema#string>	"1"^^<http://www.w3.org/2001/XMLSchema#integer>	"2019-01-05T00:00:00"^^<http://www.w3.org/2001/XMLSchema#dateTime>
"Dying To Live"^^<http://www.w3.org/2001/XMLSchema#string>	"Kodak Black"^^<http://www.w3.org/2001/XMLSchema#string>	"1"^^<http://www.w3.org/2001/XMLSchema#integer>	"2018-12-29T00:00:00"^^<http://www.w3.org/2001/XMLSchema#dateTime>
"Skins"^^<http://www.w3.org/2001/XMLSchema#string>	"XXXTENTACION"^^<http://www.w3.org/2001/XMLSchema#string>	"1"^^<http://www.w3.org/2001/XMLSchema#integer>	"2018-12-22T00:00:00"^^<http://www.w3.org/2001/XMLSchema#dateTime>
"Championships"^^<http://www.w3.org/2001/XMLSchema#string>	"Meek Mill"^^<http://www.w3.org/2001/XMLSchema#string>	"1"^^<http://www.w3.org/2001/XMLSchema#integer>	"2018-12-15T00:00:00"^^<http://www.w3.org/2001/XMLSchema#dateTime>
"Delta"^^<http://www.w3.org/2001/XMLSchema#string>	"Mumford & Sons"^^<http://www.w3.org/2001/XMLSchema#string>	"1"^^<http://www.w3.org/2001/XMLSchema#integer>	"2018-12-01T00:00:00"^^<http://www.w3.org/2001/XMLSchema#dateTime>
"Experiment"^^<http://www.w3.org/2001/XMLSchema#string>	"Kane Brown"^^<http://www.w3.org/2001/XMLSchema#string>	"1"^^<http://www.w3.org/2001/XMLSchema#integer>	"2018-11-24T00:00:00"^^<http://www.w3.org/2001/XMLSchema#dateTime>
"Not All Heroes Wear Capes"^^<http://www.w3.org/2001/XMLSchema#string>	"Metro Boomin"^^<http://www.w3.org/2001/XMLSchema#string>	"1"^^<http://www.w3.org/2001/XMLSchema#integer>	"2018-11-17T00:00:00"^^<http://www.w3.org/2001/XMLSchema#dateTime>
"I Am > I Was"^^<http://www.w3.org/2001/XMLSchema#string>	"21 Savage"^^<http://www.w3.org/2001/XMLSchema#string>	"2"^^<http://www.w3.org/2001/XMLSchema#integer>	"2019-01-19T00:00:00"^^<http://www.w3.org/2001/XMLSchema#dateTime>
"Hoodie SZN"^^<http://www.w3.org/2001/XMLSchema#string>	"A Boogie Wit da Hoodie"^^<http://www.w3.org/2001/XMLSchema#string>	"2"^^<http://www.w3.org/2001/XMLSchema#integer>	"2019-01-12T00:00:00"^^<http://www.w3.org/2001/XMLSchema#dateTime>
"Hoodie SZN"^^<http://www.w3.org/2001/XMLSchema#string>	"A Boogie Wit da Hoodie"^^<http://www.w3.org/2001/XMLSchema#string>	"2"^^<http://www.w3.org/2001/XMLSchema#integer>	"2019-01-05T00:00:00"^^<http://www.w3.org/2001/XMLSchema#dateTime>
"Championships"^^<http://www.w3.org/2001/XMLSchema#string>	"Meek Mill"^^<http://www.w3.org/2001/XMLSchema#string>	"2"^^<http://www.w3.org/2001/XMLSchema#integer>	"2018-12-29T00:00:00"^^<http://www.w3.org/2001/XMLSchema#dateTime>
"Championships"^^<http://www.w3.org/2001/XMLSchema#string>	"Meek Mill"^^<http://www.w3.org/2001/XMLSchema#string>	"2"^^<http://www.w3.org/2001/XMLSchema#integer>	"2018-12-22T00:00:00"^^<http://www.w3.org/2001/XMLSchema#dateTime>
"Street Gossip"^^<http://www.w3.org/2001/XMLSchema#string>	"Lil Baby"^^<http://www.w3.org/2001/XMLSchema#string>	"2"^^<http://www.w3.org/2001/XMLSchema#integer>	"2018-12-15T00:00:00"^^<http://www.w3.org/2001/XMLSchema#dateTime>
"DUMMY BOY"^^<http://www.w3.org/2001/XMLSchema#string>	"6ix9ine"^^<http://www.w3.org/2001/XMLSchema#string>	"2"^^<http://www.w3.org/2001/XMLSchema#integer>	"2018-12-08T00:00:00"^^<http://www.w3.org/2001/XMLSchema#dateTime>
"Love"^^<http://www.w3.org/2001/XMLSchema#string>	"Michael Buble"^^<http://www.w3.org/2001/XMLSchema#string>	"2"^^<http://www.w3.org/2001/XMLSchema#integer>	"2018-12-01T00:00:00"^^<http://www.w3.org/2001/XMLSchema#dateTime>
"Origins"^^<http://www.w3.org/2001/XMLSchema#string>	"Imagine Dragons"^^<http://www.w3.org/2001/XMLSchema#string>	"2"^^<http://www.w3.org/2001/XMLSchema#integer>	"2018-11-24T00:00:00"^^<http://www.w3.org/2001/XMLSchema#dateTime>
"Championships"^^<http://www.w3.org/2001/XMLSchema#string>	"Meek Mill"^^<http://www.w3.org/2001/XMLSchema#string>	"3"^^<http://www.w3.org/2001/XMLSchema#integer>	"2019-01-12T00:00:00"^^<http://www.w3.org/2001/XMLSchema#dateTime>
"DUMMY BOY"^^<http://www.w3.org/2001/XMLSchema#string>	"6ix9ine"^^<http://www.w3.org/2001/XMLSchema#string>	"3"^^<http://www.w3.org/2001/XMLSchema#integer>	"2018-12-15T00:00:00"^^<http://www.w3.org/2001/XMLSchema#dateTime>
"Love"^^<http://www.w3.org/2001/XMLSchema#string>	"Michael Buble"^^<http://www.w3.org/2001/XMLSchema#string>	"3"^^<http://www.w3.org/2001/XMLSchema#integer>	"2018-12-08T00:00:00"^^<http://www.w3.org/2001/XMLSchema#dateTime>
"A Love Letter To You 3"^^<http://www.w3.org/2001/XMLSchema#string>	"Trippie Redd"^^<http://www.w3.org/2001/XMLSchema#string>	"3"^^<http://www.w3.org/2001/XMLSchema#integer>	"2018-11-24T00:00:00"^^<http://www.w3.org/2001/XMLSchema#dateTime>

Figure 27: SPARQL Query #1: Results.

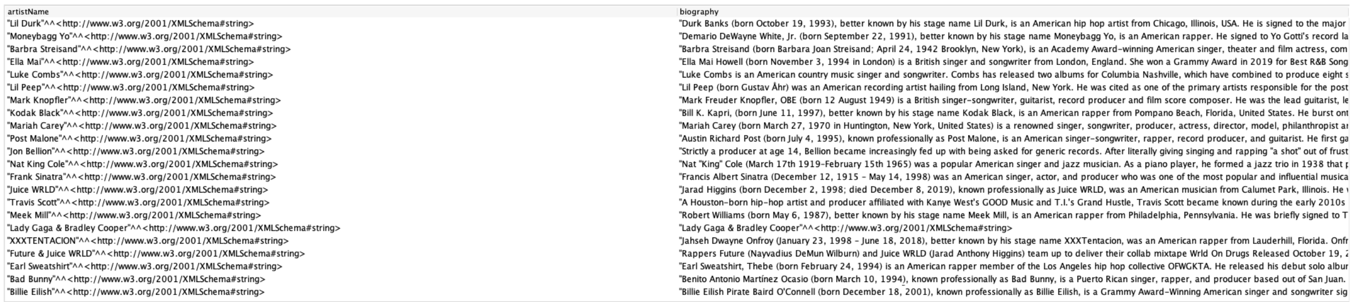
Query #2: Get number of times each album has been ranked



albumName	artistName	nrOfTimesInRank
"beerbongs & bentleys"^^<http://www.w3.org/2001/XMLSchema#string>	"Post Malone"^^<http://www.w3.org/2001/XMLSchema#string>	"10"^^<http://www.w3.org/2001/XMLSchema#integer>
"Sweetener"^^<http://www.w3.org/2001/XMLSchema#string>	"Ariana Grande"^^<http://www.w3.org/2001/XMLSchema#string>	"8"^^<http://www.w3.org/2001/XMLSchema#integer>
"Drip Harder"^^<http://www.w3.org/2001/XMLSchema#string>	"Lil Baby & Gunna"^^<http://www.w3.org/2001/XMLSchema#string>	"7"^^<http://www.w3.org/2001/XMLSchema#integer>
"Championships"^^<http://www.w3.org/2001/XMLSchema#string>	"Meek Mill"^^<http://www.w3.org/2001/XMLSchema#string>	"6"^^<http://www.w3.org/2001/XMLSchema#integer>
"Christmas Is Here!"^^<http://www.w3.org/2001/XMLSchema#string>	"Pentatonix"^^<http://www.w3.org/2001/XMLSchema#string>	"5"^^<http://www.w3.org/2001/XMLSchema#integer>
"Origins"^^<http://www.w3.org/2001/XMLSchema#string>	"Imagine Dragons"^^<http://www.w3.org/2001/XMLSchema#string>	"4"^^<http://www.w3.org/2001/XMLSchema#integer>
"Love"^^<http://www.w3.org/2001/XMLSchema#string>	"Michael Buble"^^<http://www.w3.org/2001/XMLSchema#string>	"4"^^<http://www.w3.org/2001/XMLSchema#integer>
"DUMMY BOY"^^<http://www.w3.org/2001/XMLSchema#string>	"6ix9ine"^^<http://www.w3.org/2001/XMLSchema#string>	"4"^^<http://www.w3.org/2001/XMLSchema#integer>
"A Pentatonix Christmas"^^<http://www.w3.org/2001/XMLSchema#string>	"Pentatonix"^^<http://www.w3.org/2001/XMLSchema#string>	"4"^^<http://www.w3.org/2001/XMLSchema#integer>
"Dying To Live"^^<http://www.w3.org/2001/XMLSchema#string>	"Kodak Black"^^<http://www.w3.org/2001/XMLSchema#string>	"4"^^<http://www.w3.org/2001/XMLSchema#integer>
"Not All Heroes Wear Capes"^^<http://www.w3.org/2001/XMLSchema#string>	"Metro Boomin"^^<http://www.w3.org/2001/XMLSchema#string>	"4"^^<http://www.w3.org/2001/XMLSchema#integer>
"?"^^<http://www.w3.org/2001/XMLSchema#string>	"XXXTENTACION"^^<http://www.w3.org/2001/XMLSchema#string>	"3"^^<http://www.w3.org/2001/XMLSchema#integer>
"Realer"^^<http://www.w3.org/2001/XMLSchema#string>	"YoungBoy Never Broke Again"^^<http://www.w3.org/2001/XMLSchema#string>	"3"^^<http://www.w3.org/2001/XMLSchema#integer>
"Street Gossip"^^<http://www.w3.org/2001/XMLSchema#string>	"Lil Baby"^^<http://www.w3.org/2001/XMLSchema#string>	"3"^^<http://www.w3.org/2001/XMLSchema#integer>
"A Love Letter To You 3"^^<http://www.w3.org/2001/XMLSchema#string>	"Trippie Redd"^^<http://www.w3.org/2001/XMLSchema#string>	"3"^^<http://www.w3.org/2001/XMLSchema#integer>
"Goodbye & Good Riddance"^^<http://www.w3.org/2001/XMLSchema#string>	"Juice WRLD"^^<http://www.w3.org/2001/XMLSchema#string>	"3"^^<http://www.w3.org/2001/XMLSchema#integer>
"I Am > I Was"^^<http://www.w3.org/2001/XMLSchema#string>	"21 Savage"^^<http://www.w3.org/2001/XMLSchema#string>	"3"^^<http://www.w3.org/2001/XMLSchema#integer>
"Hoodie SZN"^^<http://www.w3.org/2001/XMLSchema#string>	"A Boogie Wit da Hoodie"^^<http://www.w3.org/2001/XMLSchema#string>	"3"^^<http://www.w3.org/2001/XMLSchema#integer>
"Invasion Of Privacy"^^<http://www.w3.org/2001/XMLSchema#string>	"Cardi B"^^<http://www.w3.org/2001/XMLSchema#string>	"2"^^<http://www.w3.org/2001/XMLSchema#integer>
"Skins"^^<http://www.w3.org/2001/XMLSchema#string>	"XXXTENTACION"^^<http://www.w3.org/2001/XMLSchema#string>	"2"^^<http://www.w3.org/2001/XMLSchema#integer>
"X 100PRE"^^<http://www.w3.org/2001/XMLSchema#string>	"Bad Bunny"^^<http://www.w3.org/2001/XMLSchema#string>	"2"^^<http://www.w3.org/2001/XMLSchema#integer>
"Greatest Hits"^^<http://www.w3.org/2001/XMLSchema#string>	"Queen"^^<http://www.w3.org/2001/XMLSchema#string>	"2"^^<http://www.w3.org/2001/XMLSchema#integer>

Figure 28: SPARQL Query #2: Results.

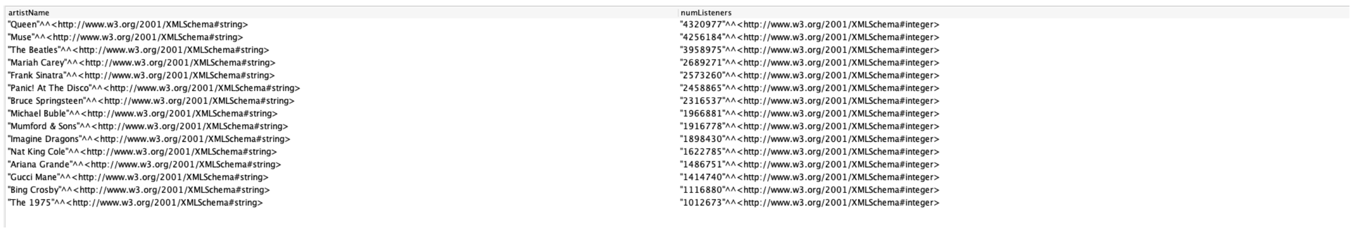
Query #3: Get Artists and their biography



artistName	biography
"Lil Durk"^^<http://www.w3.org/2001/XMLSchema#string>	"Durk Banks (born October 19, 1993), better known by his stage name Lil Durk, is an American hip hop artist from Chicago, Illinois, USA. He is signed to the major
"Moneybagg Yo"^^<http://www.w3.org/2001/XMLSchema#string>	"Demario DeWayne White, Jr. (born September 22, 1991), better known by his stage name Moneybagg Yo, is an American rapper. He signed to Yo Gotti's record la
"Barbra Streisand"^^<http://www.w3.org/2001/XMLSchema#string>	"Barbra Streisand (born Barbara Joan Streisand; April 24, 1942 Brooklyn, New York), is an Academy Award-winning American singer, theater and film actress, com
"Ella Mai"^^<http://www.w3.org/2001/XMLSchema#string>	"Ella Mai Howell (born November 3, 1994 in London) is a British singer and songwriter from London, England. She won a Grammy Award in 2019 for Best R&B Song
"Luke Combs"^^<http://www.w3.org/2001/XMLSchema#string>	"Luke Combs is an American country music singer and songwriter. Combs has released two albums for Columbia Nashville, which have combined to produce eight s
"Lil Peep"^^<http://www.w3.org/2001/XMLSchema#string>	"Lil Peep (born Gustav Åhr) was an American recording artist hailing from Long Island, New York. He was cited as one of the primary artists responsible for the post
"Mark Knopfler"^^<http://www.w3.org/2001/XMLSchema#string>	"Mark Freuder Knopfler, OBE (born 12 August 1949) is a British singer-songwriter, guitarist, record producer and film score composer. He was the lead guitarist, le
"Kodak Black"^^<http://www.w3.org/2001/XMLSchema#string>	"Bill K. Kapri, (born June 11, 1997), better known by his stage name Kodak Black, is an American rapper from Pompano Beach, Florida, United States. He burst ont
"Mariah Carey"^^<http://www.w3.org/2001/XMLSchema#string>	"Mariah Carey (born March 27, 1970 in Huntington, New York, United States) is a renowned singer, songwriter, producer, actress, director, model, philanthropist ar
"Post Malone"^^<http://www.w3.org/2001/XMLSchema#string>	"Austin Richard Post (born July 4, 1995), known professionally as Post Malone, is an American singer-songwriter, rapper, record producer, and guitarist. He first ga
"Jon Bellion"^^<http://www.w3.org/2001/XMLSchema#string>	"Strictly a producer at age 14, Bellion became increasingly fed up with being asked for generic records. After literally giving singing and rapping "a shot" out of frust
"Nat King Cole"^^<http://www.w3.org/2001/XMLSchema#string>	"Nat "King" Cole (March 17th 1919-February 15th 1965) was a popular American singer and jazz musician. As a piano player, he formed a jazz trio in 1938 that
"Frank Sinatra"^^<http://www.w3.org/2001/XMLSchema#string>	"Francis Albert Sinatra (December 12, 1915 - May 14, 1998) was an American singer, actor, and producer who was one of the most popular and influential musica
"Juice WRLD"^^<http://www.w3.org/2001/XMLSchema#string>	"Jarad Higgins (born December 2, 1998; died December 8, 2019), known professionally as Juice WRLD, was an American musician from Calumet Park, Illinois. He
"Travis Scott"^^<http://www.w3.org/2001/XMLSchema#string>	"A Houston-born hip-hop artist and producer affiliated with Kanye West's GOOD Music and T.I.'s Grand Hustle, Travis Scott became known during the early 2010s
"Meek Mill"^^<http://www.w3.org/2001/XMLSchema#string>	"Robert Williams (born May 6, 1987), better known by his stage name Meek Mill, is an American rapper from Philadelphia, Pennsylvania. He was briefly signed to T
"Lady Gaga & Bradley Cooper"^^<http://www.w3.org/2001/XMLSchema#string>	"Lady Gaga & Bradley Cooper"^^<http://www.w3.org/2001/XMLSchema#string>
"XXXTENTACION"^^<http://www.w3.org/2001/XMLSchema#string>	"Jahseh Dwayne Onfroy (January 23, 1998 - June 18, 2018), better known by his stage name XXXTentacion, was an American rapper from Lauderhill, Florida. Onfr
"Future & Juice WRLD"^^<http://www.w3.org/2001/XMLSchema#string>	"Rappers Future (Nayvadius DeMun Wilburn) and Juice WRLD (Jarad Anthony Higgins) team up to deliver their collab mixtape Wrld On Drugs Released October 19,
"Earl Sweatshirt"^^<http://www.w3.org/2001/XMLSchema#string>	"Earl Sweatshirt, Thebe (born February 24, 1994) is an American rapper member of the Los Angeles hip hop collective OFWGKTA. He released his debut solo albu
"Bad Bunny"^^<http://www.w3.org/2001/XMLSchema#string>	"Benito Antonio Martínez Ocasio (born March 10, 1994), known professionally as Bad Bunny, is a Puerto Rican singer, rapper, and producer based out of San Juan.
"Billie Eilish"^^<http://www.w3.org/2001/XMLSchema#string>	"Billie Eilish Pirate Baird O'Connell (born December 18, 2001), known professionally as Billie Eilish, is a Grammy Award-Winning American singer and songwriter sig

Figure 29: SPARQL Query #3: Results.

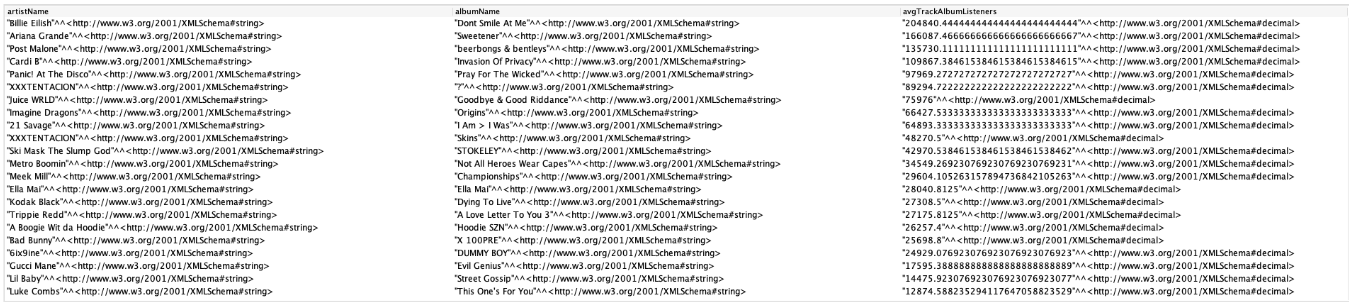
Query #4: Get 15 Artists that have more than 800000 listeners monthly



artistName	numListeners
"Queen"^^<http://www.w3.org/2001/XMLSchema#string>	"4320977"^^<http://www.w3.org/2001/XMLSchema#integer>
"Muse"^^<http://www.w3.org/2001/XMLSchema#string>	"4256184"^^<http://www.w3.org/2001/XMLSchema#integer>
"The Beatles"^^<http://www.w3.org/2001/XMLSchema#string>	"3958975"^^<http://www.w3.org/2001/XMLSchema#integer>
"Mariah Carey"^^<http://www.w3.org/2001/XMLSchema#string>	"2689271"^^<http://www.w3.org/2001/XMLSchema#integer>
"Frank Sinatra"^^<http://www.w3.org/2001/XMLSchema#string>	"2573260"^^<http://www.w3.org/2001/XMLSchema#integer>
"Panic! At The Disco"^^<http://www.w3.org/2001/XMLSchema#string>	"2458865"^^<http://www.w3.org/2001/XMLSchema#integer>
"Bruce Springsteen"^^<http://www.w3.org/2001/XMLSchema#string>	"2316537"^^<http://www.w3.org/2001/XMLSchema#integer>
"Michael Buble"^^<http://www.w3.org/2001/XMLSchema#string>	"1966881"^^<http://www.w3.org/2001/XMLSchema#integer>
"Mumford & Sons"^^<http://www.w3.org/2001/XMLSchema#string>	"1916778"^^<http://www.w3.org/2001/XMLSchema#integer>
"Imagine Dragons"^^<http://www.w3.org/2001/XMLSchema#string>	"1898430"^^<http://www.w3.org/2001/XMLSchema#integer>
"Nat King Cole"^^<http://www.w3.org/2001/XMLSchema#string>	"1622785"^^<http://www.w3.org/2001/XMLSchema#integer>
"Ariana Grande"^^<http://www.w3.org/2001/XMLSchema#string>	"1486751"^^<http://www.w3.org/2001/XMLSchema#integer>
"Gucci Mane"^^<http://www.w3.org/2001/XMLSchema#string>	"1414740"^^<http://www.w3.org/2001/XMLSchema#integer>
"Bing Crosby"^^<http://www.w3.org/2001/XMLSchema#string>	"1116880"^^<http://www.w3.org/2001/XMLSchema#integer>
"The 1975"^^<http://www.w3.org/2001/XMLSchema#string>	"1012673"^^<http://www.w3.org/2001/XMLSchema#integer>

Figure 30: SPARQL Query #4: Results.

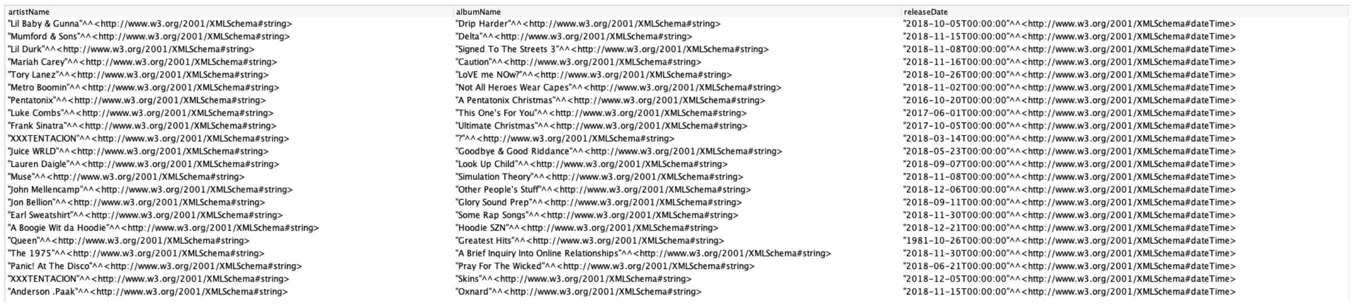
Query #5: Get the track average number of listeners by album



artistName	albumName	avgTrackAlbumListeners
"Billie Eilish"^^<http://www.w3.org/2001/XMLSchema#string>	"Dont Smile At Me"^^<http://www.w3.org/2001/XMLSchema#string>	"204840.444444444444444444444444"^^<http://www.w3.org/2001/XMLSchema#decimal>
"Ariana Grande"^^<http://www.w3.org/2001/XMLSchema#string>	"Sweetener"^^<http://www.w3.org/2001/XMLSchema#string>	"166087.466666666666666666666667"^^<http://www.w3.org/2001/XMLSchema#decimal>
"Post Malone"^^<http://www.w3.org/2001/XMLSchema#string>	"beerbongs & bentleys"^^<http://www.w3.org/2001/XMLSchema#string>	"135730.111111111111111111111111"^^<http://www.w3.org/2001/XMLSchema#decimal>
"Cardi B"^^<http://www.w3.org/2001/XMLSchema#string>	"Invasion Of Privacy"^^<http://www.w3.org/2001/XMLSchema#string>	"109867.384615384615384615384615"^^<http://www.w3.org/2001/XMLSchema#decimal>
"Panic! At The Disco"^^<http://www.w3.org/2001/XMLSchema#string>	"Pray For The Wicked"^^<http://www.w3.org/2001/XMLSchema#string>	"97969.272727272727272727272727"^^<http://www.w3.org/2001/XMLSchema#decimal>
"XXXTENTACION"^^<http://www.w3.org/2001/XMLSchema#string>	"?"^^<http://www.w3.org/2001/XMLSchema#string>	"89294.722222222222222222222222"^^<http://www.w3.org/2001/XMLSchema#decimal>
"Juice WRLD"^^<http://www.w3.org/2001/XMLSchema#string>	"Goodbye & Good Riddance"^^<http://www.w3.org/2001/XMLSchema#string>	"75976"^^<http://www.w3.org/2001/XMLSchema#decimal>
"Imagine Dragons"^^<http://www.w3.org/2001/XMLSchema#string>	"Origins"^^<http://www.w3.org/2001/XMLSchema#string>	"66427.533333333333333333333333"^^<http://www.w3.org/2001/XMLSchema#decimal>
"21 Savage"^^<http://www.w3.org/2001/XMLSchema#string>	"I Am > I Was"^^<http://www.w3.org/2001/XMLSchema#string>	"64893.333333333333333333333333"^^<http://www.w3.org/2001/XMLSchema#decimal>
"XXXTENTACION"^^<http://www.w3.org/2001/XMLSchema#string>	"Skins"^^<http://www.w3.org/2001/XMLSchema#string>	"48270.5"^^<http://www.w3.org/2001/XMLSchema#decimal>
"Ski Mask The Slump God"^^<http://www.w3.org/2001/XMLSchema#string>	"STOKELEY"^^<http://www.w3.org/2001/XMLSchema#string>	"42970.538461538461538461538462"^^<http://www.w3.org/2001/XMLSchema#decimal>
"Metro Boomin"^^<http://www.w3.org/2001/XMLSchema#string>	"Not All Heroes Wear Capes"^^<http://www.w3.org/2001/XMLSchema#string>	"34549.269230769230769230769231"^^<http://www.w3.org/2001/XMLSchema#decimal>
"Meek Mill"^^<http://www.w3.org/2001/XMLSchema#string>	"Championships"^^<http://www.w3.org/2001/XMLSchema#string>	"29604.105263157894736842105263"^^<http://www.w3.org/2001/XMLSchema#decimal>
"Ella Mai"^^<http://www.w3.org/2001/XMLSchema#string>	"Ella Mai"^^<http://www.w3.org/2001/XMLSchema#string>	"28040.8125"^^<http://www.w3.org/2001/XMLSchema#decimal>
"Kodak Black"^^<http://www.w3.org/2001/XMLSchema#string>	"Dying To Live"^^<http://www.w3.org/2001/XMLSchema#string>	"27308.5"^^<http://www.w3.org/2001/XMLSchema#decimal>
"Trippie Redd"^^<http://www.w3.org/2001/XMLSchema#string>	"A Love Letter To You 3"^^<http://www.w3.org/2001/XMLSchema#string>	"27175.8125"^^<http://www.w3.org/2001/XMLSchema#decimal>
"A Boogie Wit da Hoodie"^^<http://www.w3.org/2001/XMLSchema#string>	"Hoodie SZN"^^<http://www.w3.org/2001/XMLSchema#string>	"26257.4"^^<http://www.w3.org/2001/XMLSchema#decimal>
"Bad Bunny"^^<http://www.w3.org/2001/XMLSchema#string>	"X 100PRE"^^<http://www.w3.org/2001/XMLSchema#string>	"25698.8"^^<http://www.w3.org/2001/XMLSchema#decimal>
"6ix9ine"^^<http://www.w3.org/2001/XMLSchema#string>	"DUMMY BOY"^^<http://www.w3.org/2001/XMLSchema#string>	"24929.076923076923076923076923"^^<http://www.w3.org/2001/XMLSchema#decimal>
"Gucci Mane"^^<http://www.w3.org/2001/XMLSchema#string>	"Evil Genius"^^<http://www.w3.org/2001/XMLSchema#string>	"17595.388888888888888888888889"^^<http://www.w3.org/2001/XMLSchema#decimal>
"Lil Baby"^^<http://www.w3.org/2001/XMLSchema#string>	"Street Gossip"^^<http://www.w3.org/2001/XMLSchema#string>	"14475.923076923076923076923077"^^<http://www.w3.org/2001/XMLSchema#decimal>
"Luke Combs"^^<http://www.w3.org/2001/XMLSchema#string>	"This One's For You"^^<http://www.w3.org/2001/XMLSchema#string>	"12874.588235294117647058823529"^^<http://www.w3.org/2001/XMLSchema#decimal>

Figure 31: SPARQL Query #5: Results.

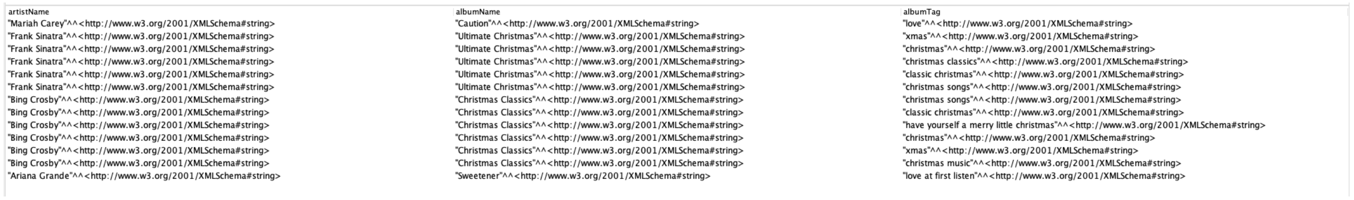
Query #6: Get Albums release date



artistName	albumName	releaseDate
"Lil Baby & Gunna"^^<http://www.w3.org/2001/XMLSchema#string>	"Drip Harder"^^<http://www.w3.org/2001/XMLSchema#string>	"2018-10-05T00:00:00"^^<http://www.w3.org/2001/XMLSchema#dateTime>
"Mumford & Sons"^^<http://www.w3.org/2001/XMLSchema#string>	"Delta"^^<http://www.w3.org/2001/XMLSchema#string>	"2018-11-15T00:00:00"^^<http://www.w3.org/2001/XMLSchema#dateTime>
"Lil Durk"^^<http://www.w3.org/2001/XMLSchema#string>	"Signed To The Streets 3"^^<http://www.w3.org/2001/XMLSchema#string>	"2018-11-08T00:00:00"^^<http://www.w3.org/2001/XMLSchema#dateTime>
"Mariah Carey"^^<http://www.w3.org/2001/XMLSchema#string>	"Caution"^^<http://www.w3.org/2001/XMLSchema#string>	"2018-11-16T00:00:00"^^<http://www.w3.org/2001/XMLSchema#dateTime>
"Tory Lanez"^^<http://www.w3.org/2001/XMLSchema#string>	"LoVE me NOw?"^^<http://www.w3.org/2001/XMLSchema#string>	"2018-10-26T00:00:00"^^<http://www.w3.org/2001/XMLSchema#dateTime>
"Metro Boomin"^^<http://www.w3.org/2001/XMLSchema#string>	"Not All Heroes Wear Capes"^^<http://www.w3.org/2001/XMLSchema#string>	"2018-11-02T00:00:00"^^<http://www.w3.org/2001/XMLSchema#dateTime>
"Pentatonix"^^<http://www.w3.org/2001/XMLSchema#string>	"A Pentatonix Christmas"^^<http://www.w3.org/2001/XMLSchema#string>	"2016-10-20T00:00:00"^^<http://www.w3.org/2001/XMLSchema#dateTime>
"Luke Combs"^^<http://www.w3.org/2001/XMLSchema#string>	"This One's For You"^^<http://www.w3.org/2001/XMLSchema#string>	"2017-06-01T00:00:00"^^<http://www.w3.org/2001/XMLSchema#dateTime>
"Frank Sinatra"^^<http://www.w3.org/2001/XMLSchema#string>	"Ultimate Christmas"^^<http://www.w3.org/2001/XMLSchema#string>	"2017-10-05T00:00:00"^^<http://www.w3.org/2001/XMLSchema#dateTime>
"XXXTENTACION"^^<http://www.w3.org/2001/XMLSchema#string>	"?"^^<http://www.w3.org/2001/XMLSchema#string>	"2018-03-14T00:00:00"^^<http://www.w3.org/2001/XMLSchema#dateTime>
"Juice WRLD"^^<http://www.w3.org/2001/XMLSchema#string>	"Goodbye & Good Riddance"^^<http://www.w3.org/2001/XMLSchema#string>	"2018-05-23T00:00:00"^^<http://www.w3.org/2001/XMLSchema#dateTime>
"Lauren Daigle"^^<http://www.w3.org/2001/XMLSchema#string>	"Look Up Child"^^<http://www.w3.org/2001/XMLSchema#string>	"2018-09-07T00:00:00"^^<http://www.w3.org/2001/XMLSchema#dateTime>
"Muse"^^<http://www.w3.org/2001/XMLSchema#string>	"Simulation Theory"^^<http://www.w3.org/2001/XMLSchema#string>	"2018-11-08T00:00:00"^^<http://www.w3.org/2001/XMLSchema#dateTime>
"John Mellencamp"^^<http://www.w3.org/2001/XMLSchema#string>	"Other People's Stuff"^^<http://www.w3.org/2001/XMLSchema#string>	"2018-12-06T00:00:00"^^<http://www.w3.org/2001/XMLSchema#dateTime>
"Jon Bellion"^^<http://www.w3.org/2001/XMLSchema#string>	"Glory Sound Prep"^^<http://www.w3.org/2001/XMLSchema#string>	"2018-09-11T00:00:00"^^<http://www.w3.org/2001/XMLSchema#dateTime>
"Earl Sweatshirt"^^<http://www.w3.org/2001/XMLSchema#string>	"Some Rap Songs"^^<http://www.w3.org/2001/XMLSchema#string>	"2018-11-30T00:00:00"^^<http://www.w3.org/2001/XMLSchema#dateTime>
"A Boogie Wit da Hoodie"^^<http://www.w3.org/2001/XMLSchema#string>	"Hoodie SZN"^^<http://www.w3.org/2001/XMLSchema#string>	"2018-12-21T00:00:00"^^<http://www.w3.org/2001/XMLSchema#dateTime>
"Queen"^^<http://www.w3.org/2001/XMLSchema#string>	"Greatest Hits"^^<http://www.w3.org/2001/XMLSchema#string>	"1981-10-26T00:00:00"^^<http://www.w3.org/2001/XMLSchema#dateTime>
"The 1975"^^<http://www.w3.org/2001/XMLSchema#string>	"A Brief Inquiry Into Online Relationships"^^<http://www.w3.org/2001/XMLSchema#string>	"2018-11-30T00:00:00"^^<http://www.w3.org/2001/XMLSchema#dateTime>
"Panic! At The Disco"^^<http://www.w3.org/2001/XMLSchema#string>	"Pray For The Wicked"^^<http://www.w3.org/2001/XMLSchema#string>	"2018-06-21T00:00:00"^^<http://www.w3.org/2001/XMLSchema#dateTime>
"XXXTENTACION"^^<http://www.w3.org/2001/XMLSchema#string>	"Skins"^^<http://www.w3.org/2001/XMLSchema#string>	"2018-12-05T00:00:00"^^<http://www.w3.org/2001/XMLSchema#dateTime>
"Anderson .Paak"^^<http://www.w3.org/2001/XMLSchema#string>	"Oxnard"^^<http://www.w3.org/2001/XMLSchema#string>	"2018-11-15T00:00:00"^^<http://www.w3.org/2001/XMLSchema#dateTime>

Figure 32: SPARQL Query #6: Results.

Query #7: Get Christmas and Love albums from american artists



artistName	albumName	albumTag
"Mariah Carey"^^<http://www.w3.org/2001/XMLSchema#string>	"Caution"^^<http://www.w3.org/2001/XMLSchema#string>	"love"^^<http://www.w3.org/2001/XMLSchema#string>
"Frank Sinatra"^^<http://www.w3.org/2001/XMLSchema#string>	"Ultimate Christmas"^^<http://www.w3.org/2001/XMLSchema#string>	"xmas"^^<http://www.w3.org/2001/XMLSchema#string>
"Frank Sinatra"^^<http://www.w3.org/2001/XMLSchema#string>	"Ultimate Christmas"^^<http://www.w3.org/2001/XMLSchema#string>	"christmas"^^<http://www.w3.org/2001/XMLSchema#string>
"Frank Sinatra"^^<http://www.w3.org/2001/XMLSchema#string>	"Ultimate Christmas"^^<http://www.w3.org/2001/XMLSchema#string>	"christmas classics"^^<http://www.w3.org/2001/XMLSchema#string>
"Frank Sinatra"^^<http://www.w3.org/2001/XMLSchema#string>	"Ultimate Christmas"^^<http://www.w3.org/2001/XMLSchema#string>	"classic christmas"^^<http://www.w3.org/2001/XMLSchema#string>
"Frank Sinatra"^^<http://www.w3.org/2001/XMLSchema#string>	"Ultimate Christmas"^^<http://www.w3.org/2001/XMLSchema#string>	"christmas songs"^^<http://www.w3.org/2001/XMLSchema#string>
"Bing Crosby"^^<http://www.w3.org/2001/XMLSchema#string>	"Christmas Classics"^^<http://www.w3.org/2001/XMLSchema#string>	"christmas songs"^^<http://www.w3.org/2001/XMLSchema#string>
"Bing Crosby"^^<http://www.w3.org/2001/XMLSchema#string>	"Christmas Classics"^^<http://www.w3.org/2001/XMLSchema#string>	"classic christmas"^^<http://www.w3.org/2001/XMLSchema#string>
"Bing Crosby"^^<http://www.w3.org/2001/XMLSchema#string>	"Christmas Classics"^^<http://www.w3.org/2001/XMLSchema#string>	"have yourself a merry little christmas"^^<http://www.w3.org/2001/XMLSchema#string>
"Bing Crosby"^^<http://www.w3.org/2001/XMLSchema#string>	"Christmas Classics"^^<http://www.w3.org/2001/XMLSchema#string>	"christmas"^^<http://www.w3.org/2001/XMLSchema#string>
"Bing Crosby"^^<http://www.w3.org/2001/XMLSchema#string>	"Christmas Classics"^^<http://www.w3.org/2001/XMLSchema#string>	"xmas"^^<http://www.w3.org/2001/XMLSchema#string>
"Bing Crosby"^^<http://www.w3.org/2001/XMLSchema#string>	"Christmas Classics"^^<http://www.w3.org/2001/XMLSchema#string>	"christmas music"^^<http://www.w3.org/2001/XMLSchema#string>
"Ariana Grande"^^<http://www.w3.org/2001/XMLSchema#string>	"Sweetener"^^<http://www.w3.org/2001/XMLSchema#string>	"love at first listen"^^<http://www.w3.org/2001/XMLSchema#string>

Figure 33: SPARQL Query #7: Results.