Department of informatics, systems and communications
Master degree in data science

Feeling the Tune: Sentiment Analysis of Sanremo 2023 Performances

DATA MANAGMENT REPORT

Mariarosaria Altieri 901736
Chiara Pilone 898697
Laura Girometti 903221

Academic Year 2022/2023

Contents

1 Introduction

The Sanremo Music Festival is a highly anticipated annual event in Italy attracting music lovers from all over the world. It is a showcase of the best and brightest talent in the music industry and a platform for emerging artists to make a name for themselves. With its rich history and cultural significance, the festival has become not only a premier event in the world of music, but a major cultural phenomenon. Each year, it attracts a massive following, with fans eagerly tuning in to watch the performances and follow the latest news and developments. The festival is a major media event, with TV and social media buzzing with discussions and opinions about the competing artists and their performances.
It is a true celebration of Italian culture and a testament to the significance of music in the lives of people around the world.
The Sanremo festival is held every February and lasts for five days. During this time, 28 singers perform their songs, which are evaluated by two juries, one composed of music experts, and another consisting of the general public.
In this project, our aim was to create a comprehensive dataset on the festival contestants by collecting information from various sources. This information includes data gathered from online magazines, which provided a commentary and rating of the songs heard by journalists in preview, Spotify data to understand the popularity of the singers, and tweets streamed during the first two nights of the festival.
The idea behind was to create an integrated database on the festival contestants, in order to gain a deeper understanding of public sentiment and opinion towards the competing artists and their performances through data analysis. This project is developed through six steps:

- *Data Acquisition*: collection of data exploiting Web scraping and API techniques,
- *Data storage*: storage of data on a persistent database,
- *Data profiling*: exploring the nature and structure of collected data,
- *Data integration and enrichment*: creation of an integrated and comprehensive database,
- *Data quality*: evaluate the different quality dimension of data,
- *Exploratory Data Analysis*: query the data in order to discover interesting aspects about our research goal.

2 Goal

The objective of our project is to create a comprehensive database that captures the rich and diverse information about the singers participating in the 2023 Sanremo Music Festival. The database will gather information from various sources, including online magazines, Spotify, and Twitter, in order to provide a broad and nuanced view of the public sentiment towards the competing artists, their pre-festival popularity, and the early assessment of their performance. Our goal is to enable a deep dive into the public's sentiment towards the songs being performed and the artists themselves, as well as to compare this sentiment with the artists' pre-existing popularity and the critical judgment of their songs.

3 Data acquisition

Interesting answers to our research question could not be provided by means of a single existing dataset; on the contrary, the problem brings out the need for consulting and merging different sources. The purpose of this phase was to identify and gather all the relevant information providers. Specifically, we focus on three primary data sources: Spotify, Twitter and reviews published by newspapers about the songs they have had the opportunity to listen in advance. The goal is to obtain the following information for each of the artists participating to the Sanremo 2023 music festival:

1. **Spotify data**, to gather data about the popularity and listening habits for the artists' songs on the platform;
2. **Twitter data**, to analyze the social media sentiment about the singers emerging after the first listening of the song they brought to the contest;
3. **Newspaper reviews**, to understand which is the opinion of music-experts journalists about the competing songs.

3.1 Spotify

Spotify is one of the world's leading music streaming platforms, providing users with access to a vast library of songs and audio content. Due to its user-friendly interface and extensive database of music, Spotify is widely used all over the world and has become a major source for music discovery and streaming.
These factors led to the choice this platform as the first data source for this project; by utilizing the Spotify API, we were able to access to the abovementioned database and collect a lot of useful information about the 28 participants in Sanremo 2023.
The Spotify API's endpoints were utilized to gather data on the 28 Italian singers. The endpoints allow access to various data and resources, and considering the goal of this project, the following information was collected:

Artist endpoint

- *Name* - The name of the artist (string)
- *Popularity* - The artist's popularity on Spotify (number)
- *Genres* - The artist's musical genres (array of strings)
- *Followers* - The number of followers the artist has on Spotify (number)

Album endpoint

- *Name* - The name of the album (string)
- *Release date* - The date the album was released

- *Number of tracks* - The total number of tracks on the album (number)
- *Average popularity* - The average popularity of the tracks on the album (number)

This last information is collected by looping over the albums and calculating the average popularity of the tracks contained in each album.
The data collected from the artist and album endpoints was combined and will be used to gain insights into the music industry and the preferences of the general public, helping us to determine who is likely to win the festival.

3.2 Newspapers

In order to supplement the information collected through Spotify API, web scraping was performed on four popular news sources: Vanity Fair, Rolling Stone, Super Guida TV and il Corriere della Sera. Since journalists are allowed to listen to the Sanremo's songs in advance, weeks before the beginning of the show (typically as part of the festival's press preview), each of them published a web article containing comments and ratings for the 28 singers.
Web scraping can be done through various tools and libraries, that allow us to extract data from websites, parse the HTML or XML code, and collect the required information, by automating the process of sending HTTP requests to a website, fetching the response and parsing the content.
The BeautifulSoup library was utilized to extract the relevant information from each page; more specifically, the following data was collected:

- *Singer*: the name of the singer who is being judged;
- *Song*: the title of the song that he/she will bring to Sanremo 2023;
- *Comment*: a brief comment about the song;
- *Rate*: a vote given by the journalist to the song (on a scale of 0 to 10).

The last information was not provided just by the Rolling Stone journal.
While for the first three journals BeautifulSoup was sufficient for extracting the information, the structure of the article published by Il Corriere della Sera led to thee need to use also the Selenium library, since the data required (singer's name, title of the song, comment, and rate) was spread across multiple pages. This made necessary to automate clicking of the "next" arrow to access each subsequent page: in fact, Beautiful Soup is used for parsing and extracting data from web pages, while Selenium is a web testing framework used for automating web browsers and interacting with web pages performing a variety of different tasks.
This web scraping activity was allowed us to gather additional information and insights

into the participants of the festival, helping to paint a more comprehensive picture of each singer and their performance in the festival while focusing on the feedback provided by a technical and unbiased audience.

3.3 Twitter

Since the strongest power in determining the Sanremo's winner is held by the public, for our project we also decided to collect in real time the tweets published during the first two nights of the show, when all the songs are performed by the artists. In this way, we will be later capable to analyze them and investigate people's sentiment about each singer.
The Twitter API v2 was used in order to stream the tweets, focusing exclusively on those written in Italian language and containing the official hashtag of the event, "Sanremo2023".
Due to the huge amount of tweets that will reasonably be produced during the show, instead of directly streaming and storing the data we decided to rely on Kafka, a distributed, scalable, and fault-tolerant publish-subscribe messaging system that is designed to handle real-time data streams. In order to ensure that the Kafka cluster is scalable, robust and easy to manage, we decided to run it in Docker containers. This allows us to package and isolate the components of the Kafka cluster, such as the broker, zookeeper, and producer/consumer, as separate containers, making it easier to deploy and manage the cluster. By utilizing Docker, we can ensure that the Kafka cluster can run consistently and reliably, even if the host system experiences changes or downtime.
By utilizing Apache Kafka in conjunction with our Twitter Stream, we are capable to achieve a more robust and scalable architecture for data collection and processing. Indeed, instead of having a single module in Python that is responsible for collecting, processing and storing the data, the architecture is separated into two components: a "Producer" and a "Consumer".
The "Producer" component is responsible for collecting the data from the Twitter Stream and writing it to the Kafka queue as raw logs without any processing, while the "Consumer" component then reads the logs from the Kafka queue, performs the necessary processing and persists the data to the storage system. By implementing this architecture, the responsibility of collecting and processing the data is separated, making the system more fault-tolerant. The "Producer" can continue to collect data and write it to the queue, even if the "Consumer" experiences errors or downtime, ensuring that data is not lost. Additionally, this decoupled architecture makes it easier to scale the processing component, as the "Consumer" can be run on multiple nodes, processing the logs in parallel, to handle increased load.
Here there is a high level overview of what the data flow will look like:

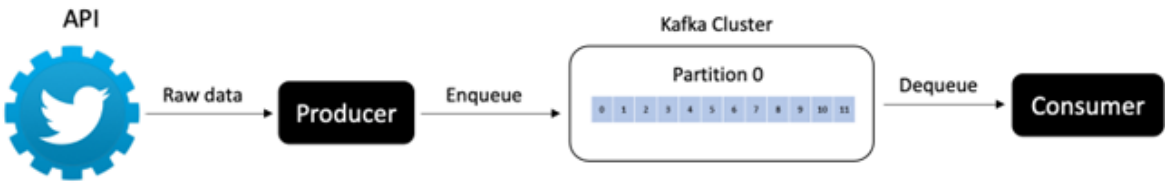

Figure 1: Data flow overview

The producer module streams the tweets to a specified Kafka topic, a named stream of records within the cluster. The records, in this case, are the tweets. The consumer module, on the other hand, consumes the data from the topic by reading the records and extracting, for each tweet:

- *Text*: the content of the tweet;
- *Date*: the time of publication of the tweet.

The consumer is also concerned with the data cleaning; since tweets may contain a lot of useless information, a function is used in order to remove from the text all the mentions and the urls, while emojis are kept in order to better understand the sentiment expressed in the tweets; indeed, emoticons play a crucial role in conveying emotions and sentiments, thus their preservation in the text before analysis is deemed necessary to arrive at more accurate results.
This processed data is then stored in MongoDB.

4 Data Storage and architecture

For our project - as partially mentioned before - we decided to use MongoDB to store the data collected during the acquisition phase. MongoDB is a NoSQL database that uses a document-based data model; this allows for the flexible storage of data in a format that can be easily updated, searched and processed, and flexibility is a fundamental characteristic when it comes to storing large volumes of data that come from different sources, like in this specific case. In fact, we collected data through Spotify API, Twitter API and web scraping, which implies a relevant variety of data structures and formats. MongoDB can handle this type of unstructured data, allowing us to store and manage it in a more efficient manner.

In MongoDB, each document in a collection can have a different structure (schemaless), allowing for easy modeling of complex relationships between data. For example, an artist may have different genres associated with him, which can be easily stored as an array in the document without having to create additional tables or columns in a relational database.

In our data collection process we determined that a document-based NoSQL database would be the most fitting for our needs. in the case of the Twitter and Spotify APIs, the responses to API requests are typically returned in JSON format, which can be easily modeled in Python as lists and dictionaries. This structure allows for easy manipulation and storage of the data, making a document-based NoSQL database such as MongoDB a natural choice for data storage.

Additionally, MongoDB is known for its scalability, high performance, and flexible data modeling capabilities, making it an ideal choice for storing large volumes of tweets, which can grow very rapidly. It also offers easy indexing and search options, which can be useful in retrieving tweets based on specific criteria.
In conclusion, MongoDB provides a flexible, scalable, and powerful solution for storing and processing large amounts of data from multiple sources, which is why it can be a good choice for your our specifc case.
The Python library "Pymongo" was used in order to interact with the database, creating three different collections to store the data retrieved during the acquisition step. We will have a collection storing the data about the artists coming from Spotify, a second collection where each document is a critic made by a journalist about a specific artist, and a third collection containing the cleaned tweets. We will use them as the basis for the integration phase, that will bring us to a fourth, final collection, highly nested, where each document will contain all the relevant information about a singer: his or her music general information, the reviews made about him (in a list of 4 json documents, one for each journal) and the tweets that refer to him (in another collection of json documents, one for each tweet). This "embedding" approach of the document based dbms, allows us to store all the relevant information about a singer in

a single document, making it easier and much more efficient to manage and analyze the data of artists. Indeed, in MongoDB, embedding is often the preferred method when it comes to maintaining relationships between collections, as it avoids the overhead of joins and reduces the amount of round trips to the database.

5 Data Profiling

After acquiring the data it is essential to conduct a preliminary examination of the data to understand its properties and characteristics. This is where data profiling comes into play: this activity provided a quick overview of the content, allowing us to identify any potential issues or inconsistencies that need to be addressed before integrating the data.
With regards to the collection of newspaper data, we have calculated the main summary measures (mean, mode, and median) of the different news outlets and represented their distribution through box plots.

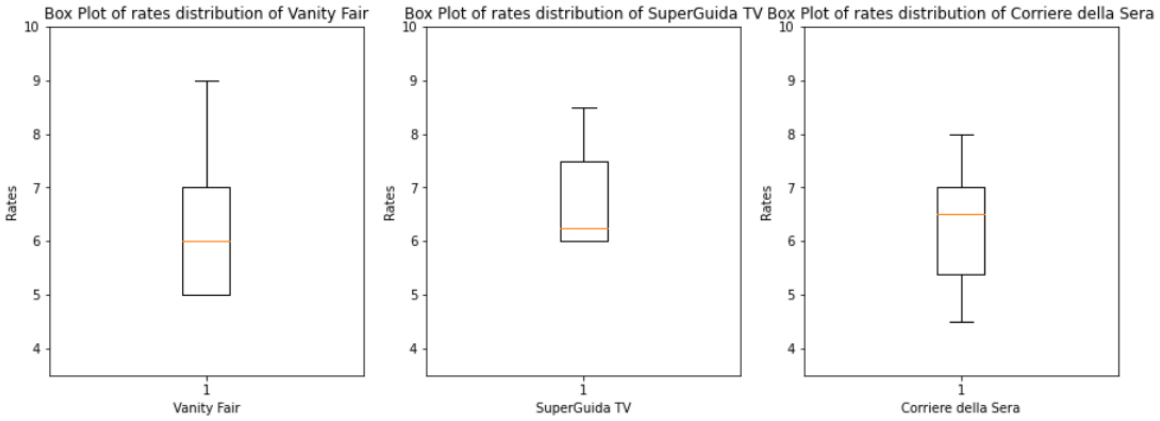

Figure 2: Distribution of newspapers' rates

From the results we can immediately see that "Corriere della Sera" is the one that gave higher median values to the artists' songs. A closer examination of the related boxplot reveals a strong negative skewness, suggesting that some songs received particularly low ratings, even reaching the insufficiency. On the other hand, we note how "SuperGuidaTv" turns out to have been the only one that assigned all singers a value of six or higher. "Vanity Fair" was the only publication to assign a rating of 9 to a song, specifically to the song of Colapesce Dimartino.
As previously mentioned, it is evident from the results that there is a missing evaluation of the songs in the competition in the Rolling Stones newspaper. Additionally, a review of the competitors shows that they are cited differently in different newspapers, with different spellings being used to refer to the same person (for example, "Paola e Chiara" is also written as "Paola & Chiara" or "Ariete" and "ARIETE"). This may pose challenges in integrating the different sources, and we will see later on how these cases have been handled.

Regarding the "arstist" collection, we can take a brief examining the artists who have gained the most popularity among the public according to Spotify. To represent this information visually, we can create a bar chart that displays the ten most popular artists. This chart can provide insight into the popularity of each artist relative to one another.

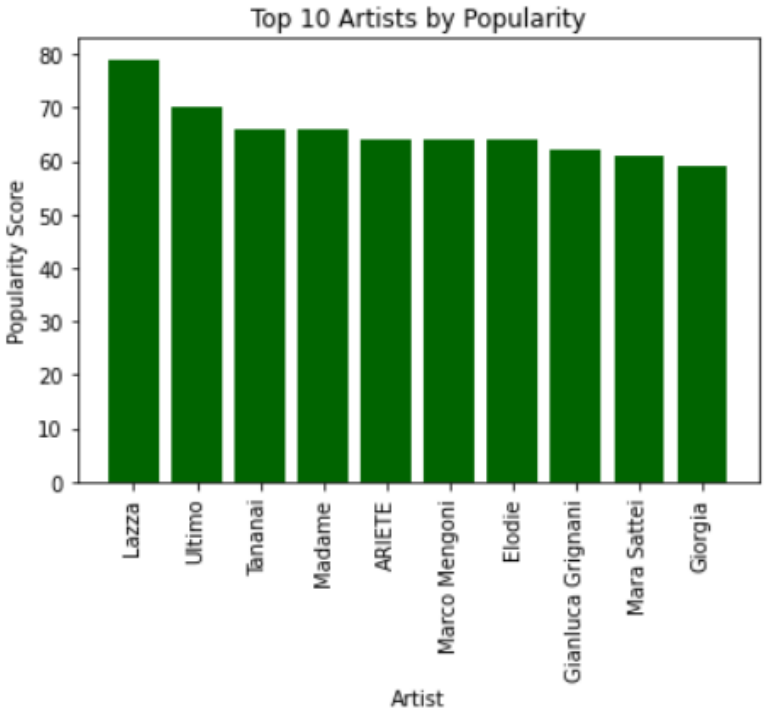

Figure 3: Top 10 artist by Spotify popularity

Additionally, to gain a better understanding of each artist's presence in the music industry, we can document the number of albums produced by each artist. This information can serve as a key indicator of an artist's length of time in the industry and can provide us with valuable insight into whether they are established, well-known figures or if they are relatively new discoveries. A higher number of produced albums generally indicates a longer period of active involvement in the music industry.

Singer	Number of Albums
Anna Oxa	29
I Cugini Di Campagna	29
Gianluca Grignani	22
Giorgia	17
Marco Mengoni	16
Paola & Chiara	12
Modà	10
Articolo 31	10
Lazza	8
Levante	8
Colapesce	7
Ultimo	5
Mr.Rain	4
Coma_Cose	3
Elodie	3
Rosa Chemical	3
Olly	2
gIANMARIA	2
Tananai	1
ARIETE	1
Madame	1
LDA	1
Mara Sattei	1
Leo Gassmann	1
Sethu	0
Shari	0
Will	0
Colla Zio	0

Figure 4: Number of album for artist

6 Data integration and enrichment

At this point, we discuss the process of integrating the three separate collections into a single, integrated database system. The purpose of this integration was to simplify data management and increase efficiency by allowing users to access information from all three sources in a centralized location.
We used the binary composed approach to integrate different collections, in fact we started from the first two collections, "artists" and "reviews", and we produce the first integrated schema, that will be further integrate with other collection in order to produce the final result.
The first step was to associate each singer with the reviews published about him in different journals.
Is important to underline that data integration usually have three different steps:

1. **Schema Transformation** → It refers to the process of converting data from one schema (data structure) to another schema in order to integrate data from different sources into a common structure;

2. **Schema Matching** → It's the task of identifying correspondences between elements of two or more schemas. This is an important step in data integration to ensure that data from different sources can be integrated seamlessly;

3. **Schema Integration** → It's the process of combining multiple schemas into a single, unified schema that can be used to store integrated data from multiple sources. This involves schema matching and schema transformation to ensure that all data fits into the same structure.

However, we have done only the last two steps because we already had the same data model for the three different datasets.
We started to figure out the correspondences between the two schemas and after observing them, we discovered that there was a common key "singer" that could be exploited in order to execute the merge.
The problem we faced was that the values associated with the "singer" key, the name of the artist, could be written differently in the two collections (e.g. Paola e Chiara , Paola Chiara). To overcome this issue, we have used Levenshtein's distance method (also known as edit distance) to measure the difference between two strings of characters. This concept measures the minimum number of single-character edits required to transform one string into the other. This concept is useful in the merging process, as it allows for the comparison of string values even if they are not exactly identical, taking into account minor differences such as typos or variations in spelling. In our case, we utilized the Python library "fuzzywuzzy" and its function " fuzz.ratio " that return a score based on Levenshtein's distance returns a score that is based on the Levenshtein distance between the two strings, which measures the number of edits

(insertions, deletions, and substitutions) needed to change one string into another. The score returned by fuzz.ratio ranges from 0 to 100, with 100 indicating an exact match between the two strings.
We fixed a threshold to associate the respective review with the particular singer and used a list in which we inserted all documents that referred to that singer. Therefore, we decided that if the value of the distance is greater than 80, the strings are considered equal.
As a result, the integrated schema includes a new key "reviews" where the value is the list of dictionaries associated with the singer, one for each review made for the artist. By using this method we were able to successfully match the correct reviews with the correct singers in the integrated schema, leading to a more comprehensive and useful database.
After obtaining this schema, we proceeded to integrate to it the tweets of the singers. What we wanted to do was to associate each singer with the tweets that mentioned him. To achieve this, we leveraged the information contained in the "tweets" collection. To identify the tweets that referred to a specific singer, we employed a process to find out the relevant tweets. It is possible that a tweet could refer to zero, one, or multiple singers. In the case where a tweet did not refer to any singer, it was what we call "junk tweet."
Our approach was to associate each singer with a list containing all the tweets referring to him. However, first we had to find a way to figure out who each tweet was referring to.
To do this we have used the Named Entity Recognition (NER) that is a sub-task of Natural Language Processing (NLP) that involves identifying and extracting entities from unstructured text. Entities can be people, organizations, locations, dates, and more. We have applied this function, provided by spacy library, on tweets and so we have extracted the entities that represent people from the text of tweets. Since a group was not recognise as a person we included also the category "organizations". For each tweet the recognized entities were collected into a list later associated to a new key called "reference".
Subsequently, we faced the challenge of identifying tweets about a singer even when the name was not mentioned explicitly. To resolve this issue, we created a dictionary that linked each singer with a list of all possible names, nicknames, and stage names they may have been referred to in the tweets.
With this dictionary in place, we merged the collection that contained the Spotify data and reviews, with the one that contained tweets. In this case a different distance function was used: token set ratio (provided by the python library fuzzywuzzy). This metric computes the ratio of the number of unique tokens (e.g., words) in the intersection of two sequences to the number of unique tokens in the union of the sequences. This approach can be useful when you want to find matches based on the

similarity of the distinct tokens or words they contain, rather than just their overall similarity as sequences.

For our use case, we believed that the fuzz.token_set_ratio could be more appropriate than the function used above. Indeed it takes into account the different variations of the "singer" name, e.g. "Marco Mengoni", "mengoni", "marco", and "marco mengoni" are all different sequences but they have the same unique tokens, so a high token set ratio will indicate that they refer to the same entity.

After having tested this function on a sample of tweets, we discovered that a threshold equal to 80 could be appropriate. Therefore, we added the tweets exceeding this threshold to a list, which became the value of the "tweets" key associated to each singer.

To avoid that the same tweet was associated to a singer multiple times, we have done a check before the append operation. For example, in the case of "Paola e Chiara", the previous function recognized them as two distinct individuals. As a result, the same tweet about both singers was mapped to the same key twice. This prior control allowed us to manage this situation. Therefore in our final collection we have a list of 28 documents, one per singer. Each document contains all the data coming from Spotify API, all the reviews published about the artist stored in an array associated to key "revies" and all the tweets referencing to him. We decide to use the embedding approach. In fact this allow us to have all relevant information about each singer in one document, avoiding in this way to perform expensive aggregations among different collections. However, if the number of tweets collected for each singer was significantly higher, for example if the stream was performed during all nights of the show, a referencing approach could be better since the maximum dimension allowed for a document (16Mb) would be exceeded. We know that the data integration process may lead to some errors; in our specific case, two different types of error may happen: we can associate a tweet to a singer, even if the tweet is not talking about him, or we can not recognise that a given tweet is talking about the singer. As regards the first error, we have tried to address all the different ways in which an user may refer to a given singer, but many other options that we did not consider exist. In other cases, the algorithm used to recognise entities in the tweets' text does not work properly. While this type of error is more difficult to calculate we tried to give an estimate of the second one; we took a sample of 100 tweets and checked manually if each tweets is correctly referencing to the singer whom he is associated. We obtained an error of 7%; we found some situations where a tweet was inserted inside the list of tweets of a given singer, even if in its text there was no intent to talk about him. For example, in the array of tweets embedded in the "Articolo 31" document, we found a tweet saying: "Benigni ha detto molte cose. *Articolo 11*,lo ha ribadito più volte!". It is clear that this tweet was not referring in any way to the singers, but the function of distance used to compute the match is not capable to understand it.

Figure 5: Final collection structure

7 Data quality

Data quality represents a crucial aspect of this process. It is important because it directly impacts the accuracy and reliability of decisions made based on that data. Poor data quality can lead to incorrect analysis, flawed conclusions, and sub-optimal outcomes. On the other hand, high-quality data can provide a solid foundation for informed decision making and improve the chances of success. Data quality refers to the level of excellence of data in terms of accuracy, completeness, consistency and currency with respect to the purpose of the data collection.
Furthermore, as we know the data integration phase leads to numerous data quality issues, it could be interesting to understand the coverage with which the observed phenomenon is represented in the data set. In our case this is very difficult, because we don't know the total number of tweets that were created during the first two nights of the show. In fact, we have collected about 300.000 tweets and this could be enough to reach our goal - and so, to get an idea about the general sentiment of twitter users about singers and their songs - but sure, many other tweets were produced, that our stream was not able to capture; many other tweets were published without the hashtag we were looking for; many other comments were provided on different platforms. So we can conclude that, when we talk about tweets data, we can not reach absolute completeness, but we can assume that the data collected is a representative sample of the phenomenon. Moreover, just a third of the 300.000 collected tweets were associated with the singers. This result it is quite plausible: we have to consider that by drawing from a data source like Twitter, it is very easy to collect a lot of very general information that does not fully fall within our domain of analysis. Mainly the reasons for such a low association of tweets to the respective singers can be attributed to several factors. One reason is that Twitter users tend to focus on the most impactful events of the evening (such as the presence of Chiara Ferragni or other special guests or unplanned event as the Blanco's performace). This can result in the artists' performance being overshadowed. Additionally, as users comments in real-time, they may not always refer to the artists by name, but instead express their opinions directly, making it harder to associate the tweets with the artists. Another reason is the moderate accuracy of the matching between the collection of artists and tweets. The current integration methodology is based on the artist's name using NER methodology and a distance function. However,in some tweets, the NLP algorithm may not always recognize artists as entities proactively excluding the tweet from the association.
To improve the quality of the data and increase coverage, it is recommended to revise the matching rules and supplement the current methodology with another system. For example, incorporating an additional system that takes into consideration the time the tweet was posted, along with the use of the actual scheduling of the evenings.

This information, which will be available after the festival has taken place, can serve as a supplementary source to increase the reliability and coverage of our data. By combining the existing methodology with this new system, we can expect to achieve a more comprehensive and precise association between the tweets and respective artists. In conclusion of this step, we also decided to get a measure of completeness for the collection "reviews", calculated after having explored the structure and nature of the data collected. In particular, we calculated the attribute completeness, i.e. the presence of non-null values for the attribute rate. We obtained a completeness equal to 75%, as one of the four magazines had not assigned any ratings to the songs listened to.

8 Exploratory data analysis

After integrated the data in a unique database we conducted an Exploratory Data Analysis (EDA) to get an overview of the information we've collected. This step in the analysis process aimed to uncover patterns, relationships and trends in the information, by utilizing techniques such as visualization, statistical analysis and summarization. EDA helps us gain a deeper understanding of the data and identify any potential outliers, anomalies, or missing values.
We initially conducted a review of the number of tweets related to each singer and discovered that some artists have a remarkably low number of tweets, with some not even reaching 1,000 tweets (such as Shari). This can largely be attributed to the frequent twitter down that occurred during the evening, which inhibited users from posting and viewing tweets. In fact, in general, artists who performed on the second night generally have fewer tweets.

8.1 Sentiment analysis

One of the main goal of our project is to understand the sentiment of people about Sanremo's songs, since public plays a major role in determining the winner of the competition. In order to achieve this objective, we wrote a function that performs the following steps:

- *Punctuation removal and text normalization*: The first step removes all punctuation from the text and converts it to lowercase.
- *Stopword removal*: Stopwords are common words that are typically filtered out in NLP tasks as they don't contribute much to the meaning of the text. In this case, the function uses the Italian stopwords from the NLTK library.
- *Word stemming*: Word stemming is the process of reducing words to their base or root form. In this case, the function uses the Italian SnowballStemmer from the NLTK library.
- *Sentiment analysis*: The final step uses the Vader Sentiment Intensity Analyzer to perform sentiment analysis on the preprocessed text. The analyzer returns a sentiment score for the text, which is a compound score that ranges from -1 to 1, with -1 being extremely negative, 1 being extremely positive, and 0 being neutral.

By applying this function to the text of each tweet associated to a given singer, and computing the average, we obtained the sentiment for each participant. This brought us to acknowledge that, respectively, Tananai, Mara Sattei and Marco Mengoni are the three singer with the highest average sentiment. However, we have to remember

that these results take into account just the tweets published during the first 2 night of the festival, and opinions may change over the course of the event, after several listening of the songs.

8.2 Queries in MongoDB

In order to make further analysis on the integrated data set, we developed some queries by using the python library Pymongo.
First of all, for the three singers with the highest average sentiment we extract both the popularity attributed by Spotify and an average rate given by the magazines, obtaining the following result:

```
Colapesce popularity =  52  average_rate =  7.333333333333333
Mara Sattei popularity =  61  average_rate =  6.5
Marco Mengoni popularity =  64  average_rate =  7.833333333333333
```

Figure 6: Queries

Then we tied to imagine some interesting queries that could be performed on the data, for example we calculate the average rate given by journalists for a specific singer, Giorgia. In the last query we found out which is the album of Marco Mengoni with the highest popularity, called MATERIA (TERRA). This queries could be performed to find interesting things also about other singers by changing only the name associated to the key singer in the query.

9 Conclusion

In conclusion, this project puts us in a real world situation in the context of data management, with all the relative problems and difficulties that exist in this field and that we tried to address. Indeed, no absolute and perfect solution exists, but we must try to seek for the right trade-offs, keeping in mind the goals of our task and the possible use-case.
The practical development of the project has made us understand that some improvements could be done in order to achieve even better results; for example, as regards the faced issue according to which many tweets do not refer explicitly to a singer, we could use the time of the performance in order to try to understand who a tweet is referred to.
Moreover, as the number of tweets grows, it may be a good idea to perform the nlp algorithm on the tweets' texts and the matching algorithm by using a distributed system, in order to exploit the processing power of different computers by using, for example, Apache Spark. In this way, the operations of extracting entities from the text and associating a tweet to a singer could be performed in parallel on different nodes, rather than in serial way. However, this makes sense just when the number of tweets to process increases significantly; if the data is low, the introduction of a distributed system would bring some issues (like the need of managing the communication between the different nodes) whose disadvantages would probably be bigger than those that we would face by performing the tasks on a single machine.
Thinking about the results we obtained, and comparing them with the results shown by the first 2 rankings presented during the show, we can assume that since 2 of the singers who have the highest average public sentiment (Tananai and Marco Mengoni) are also in the top 3 provided by the pressrom, probably here is hidden the next winner of the 73rd edition of the Sanremo music festival!