Spark-y Rock Anthem

Zaid (zp2090), Khalid (ka2612), Mahasmrti (ma7478), Prajna (pn2224), Niegil (nm4075)

ABSTRACT

The project aims to analyze music sales and popularity using Apache Spark to gain insights into which genres and artists are most popular, as well as which factors contribute to the success of a particular track or album. The project also includes features such as searching for songs based on their similarity to a given query song using MinHash LSH and generating lyrics using GPT-2. The data for the project was obtained by scraping data from Rate Your Music API, Spotify API, and Genius API, which includes information on various features of each song, such as tempo, loudness, and key, as well as metadata such as artist name, album name, and genre. Apache Spark was used to analyze this data and implement the song search and lyrics generation features. Overall, the project aims to provide valuable insights into the music industry for music industry professionals, artists, and consumers alike.

INTRODUCTION

The music industry has undergone significant changes over the past few decades, with the rise of digital music sales and streaming services changing the way consumers access and listen to music. As a result, there is a growing need for data-driven insights that can help music industry professionals understand which genres and artists are most popular, as well as which factors contribute to the success of a particular track or album.

To address this need, this project aims to analyze music sales and popularity using Apache Spark. Specifically, we will be analyzing data related to music tracks, albums, and artists, including factors such as sales, popularity, and genre. By analyzing this data, we hope to gain insights into which genres and artists are most popular, as well as which factors contribute to the success of a particular track or album.

In addition to analyzing music sales and popularity, we have also implemented features that allow users to search for songs based on their similarity to a given query song using MinHash LSH, as well as generate lyrics using GPT-2. These features could be useful for music enthusiasts who are looking for new songs to listen to or who want to explore new genres.

To obtain the data for this project, we scraped data from Rate Your Music API, Spotify API, and Genius API. The scraped data includes information on various features of each song, such as tempo, loudness, and key, as well as metadata such as artist name, album name, and genre. Additionally, we scraped lyrics for each song from the Genius API.

To analyze this data and implement the song search and lyrics generation features, we used Apache Spark, a distributed computing system that allows for the processing of large datasets in a parallel and fault-tolerant manner. Specifically, we used Spark's DataFrame API to manipulate and analyze the data, as well as Spark's implementation of MinHash LSH for the song search feature and GPT-2 for the lyrics generation feature. We used python, a high-level programming language that is well-suited for distributed computing and data analysis.

Overall, this project aims to provide valuable insights into the music industry by analyzing data related to music sales and popularity. The insights gained from this analysis could be useful for music industry professionals, artists, and consumers alike.

METHODOLOGY

Data Preparation

The first step in our methodology was to collect data from various sources, including Rate Your Music API, Spotify API, and Genius API. We gathered information on over 800,000 songs, including metadata such as artist name, album name, and genre, as well as features such as tempo, loudness, and key. Additionally, we scraped lyrics for each song from the Genius API, resulting in a dataset of 646,480 lyrics.

The data covers a wide range of years from 2000 to 2023, with a total of 17 GB of data. The data includes information on 5 music charts and 1,040 albums. There are 812,577 unique songs and 3,104 genres in the dataset.

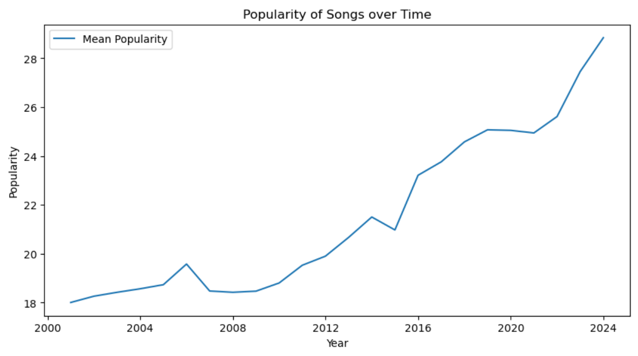
To prepare the data for analysis, we performed several preprocessing steps, including:

- **Removing duplicates**: We removed any duplicate records that may have been present in the scraped data.
- **Handling missing values**: We checked for and handled any missing values in the dataset. In some cases, we input missing values using mean or median values.
- **Data transformation**: We transformed the data as needed to prepare it for analysis. For example, we converted data types to the appropriate format, and we merged multiple datasets to create a comprehensive dataset for analysis.
- **Filtering and aggregation**: We filtered the data to remove any outliers or irrelevant records. We also performed aggregation on the data to group it by relevant attributes such as artist, album, or genre.
- **Feature engineering**: We created new features based on the existing data. For example, we created a popularity score for each song based on its Spotify popularity and other factors.

After performing these preprocessing steps, we were left with a clean and comprehensive data set for analysis. The next step was to use Apache Spark to analyze the data and gain insights into the music industry.

EDA:

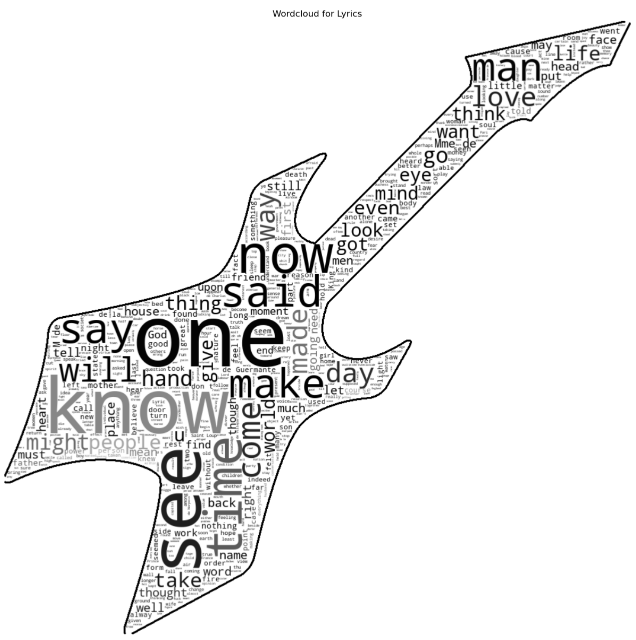
Popularity of Songs over Time

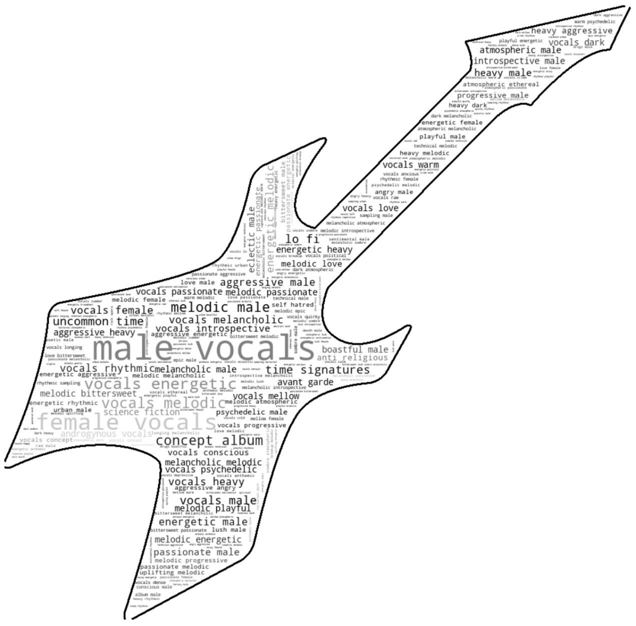


It can be observed in the visualization above that the popularity of songs have been steadily increasing over the years because of several reasons. One of the most significant reasons include the accessibility offered by online music platforms like Spotify, which enables listeners to find new music from all around the world. These platforms' personalization algorithms also make song recommendations based on user preferences. Social media has made it easier for people to share their favorite songs and playlists, which has helped to create a viral impact that can swiftly boost a song's popularity. The growing interconnectedness of the world has also given musicians from various nations and cultures access to international audiences, enhancing the variety of music and the opportunity for musicians to become well-known.

Word Clouds

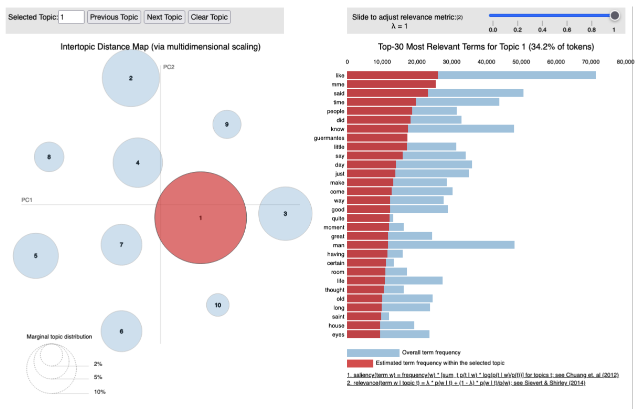
The first wordcloud is generated based on the frequency of words in the song lyrics. Love, Hand, Think, Want, Eye, One, Know, See, Say, Time, Come, Make, Day are some of the highest occurring words in the dataset. These words may relate to Love songs(Love, Hand, Eye, Man), expressing experience (know, say, see) amongst many other things.





Let's talkaobut the second wordcloud. Each song when scraped came with its own list of descriptors. The word cloud resembles the frequently occurring descriptors in the dataset. “Male vocals”, “melodic male”, “melancholic” seem to be the one repeating the most.

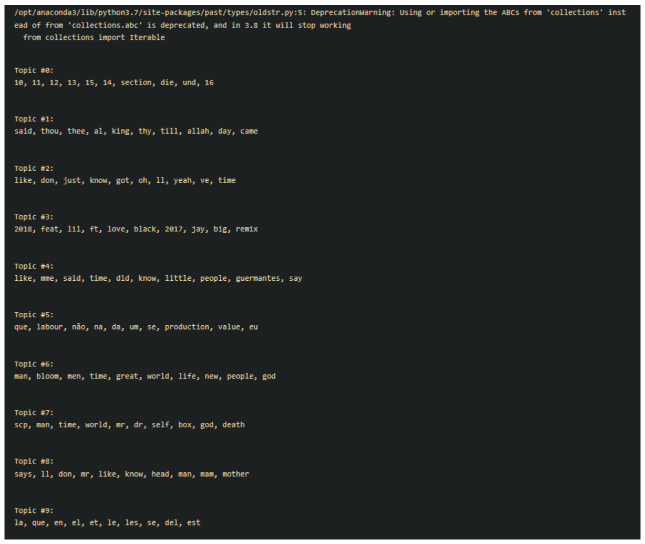
Topic Modeling using LDA


Selected Topic: 1
Previous Topic
Next Topic
Clear Topic
Slide to adjust relevance metric:(2)
λ = 1
0.0
0.2
0.4
0.6
0.8
1
Intertopic Distance Map (via multidimensional scaling)
PC2
PC1
1
2
3
4
5
6
7
8
9
10
Marginal topic distribution
2%
5%
10%
Top-30 Most Relevant Terms for Topic 1 (34.2% of tokens)
0
10,000
20,000
30,000
40,000
50,000
60,000
70,000
80,000
like
mme
said
time
people
did
know
guermantes
little
say
day
just
make
come
way
good
quite
moment
great
man
having
certain
room
life
thought
old
long
saint
house
eyes
Overall term frequency
Estimated term frequency within the selected topic
1. saliency(term w) = frequency(w) * [sum_t p(t | w) * log(p(t | w)/p(t))] for topics t; see Chuang et. al (2012)
2. relevance(term w | topic t) = λ * p(w | t) + (1 - λ) * p(w | t)/p(w); see Sievert & Shirley (2014)

Latent Dirichlet Allocation (LDA) is a statistical model used in topic modeling and natural language processing in machine learning. Finding hidden themes or subjects in a group of papers is the objective of LDA. By definition, according to the LDA model, each document in the corpus is made up of a variety of themes, and each topic is a distribution across a predetermined set of words. The topic distribution for each document and the word distribution for each topic are determined by the model using a probabilistic technique. Here in our case, LDA helped us find various underlying topics in the song lyrics.

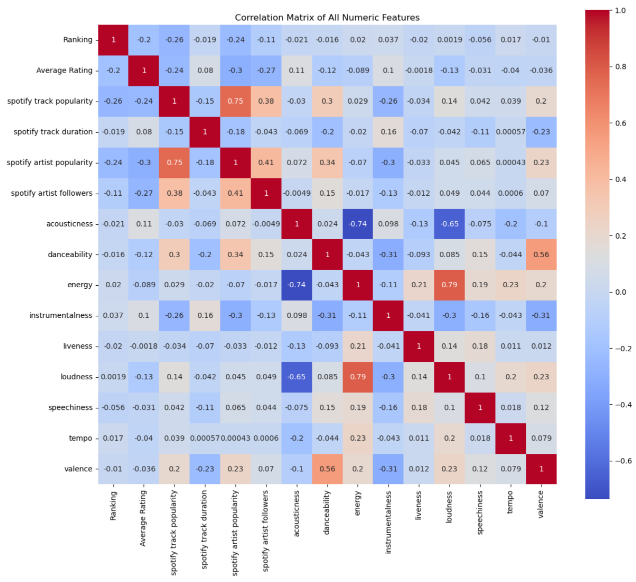
The visualization above has the frequently occurring relevant terms for that particular topic. This topic appears to have words related to religion or mythology, with words like "thou", "thee", "king", "thy", and "allah".

Pertaining to the results in the following visualization, for topic #6, the phrases "man", "bloom", "men", "time", "great", "world", "life", "new", "people", and "god" suggest that this topic is mostly about people, life, and possibly philosophical ideas whereas topic #9 suggests that the majority of the words in this topic are foreign terms, possibly from French and Spanish, including "la," "que," "en," "el," "et," "le," "les," "se," "del," and "est."



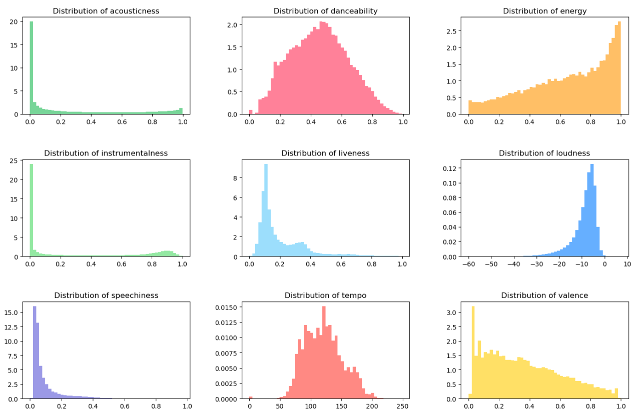
```
/opt/anaconda3/lib/python3.7/site-packages/past/types/oldstr.py:5: DeprecationWarning: Using or importing the ABCs from 'collections' inst
ead of from 'collections.abc' is deprecated, and in 3.8 it will stop working
  from collections import Iterable


Topic #0:
10, 11, 12, 13, 15, 14, section, die, und, 16


Topic #1:
said, thou, thee, al, king, thy, till, allah, day, came


Topic #2:
like, don, just, know, got, oh, ll, yeah, ve, time


Topic #3:
2018, feat, lil, ft, love, black, 2017, jay, big, remix


Topic #4:
like, mme, said, time, did, know, little, people, guermantes, say


Topic #5:
que, labour, não, na, da, um, se, production, value, eu


Topic #6:
man, bloom, men, time, great, world, life, new, people, god


Topic #7:
scp, man, time, world, mr, dr, self, box, god, death


Topic #8:
says, ll, don, mr, like, know, head, man, mam, mother


Topic #9:
la, que, en, el, et, le, les, se, del, est
```

Correlation between Numerical Song Features



The above correlation matrix shows some interesting relations between the various numerical song features. Energy and loudness have a very significant positive association (0.790280), meaning that as a track's energy rises, so does its volume. Between acousticness and energy (-0.736476) as well as between acousticness and loudness (-0.651035), there is a significant negative association. This shows that a track's intensity and loudness diminish as its acoustic quality increases. According to a high positive correlation between Spotify track and artist popularity (0.751289), more well-known musicians typically have more well-liked tracks. Danceability and valence have a weakly positive connection (0.557238). This suggests that tunes with more danceability tend to have emotional content that is more upbeat (happy). Danceability and valence are moderately negatively correlated with instrumentalness (-0.313365 and -0.311694, respectively). This implies that songs with greater instrumental content typically have a lower valence and are less danceable. Ranking and Average Rating (-0.198654) as well as Ranking and Spotify Track Popularity (-0.258534) have a marginally negative association. According to this, tracks with higher rankings (lower numerical values) typically have higher average ratings and greater popularity on Spotify, but the correlation between the two is not very strong.

Distribution of Audio Features

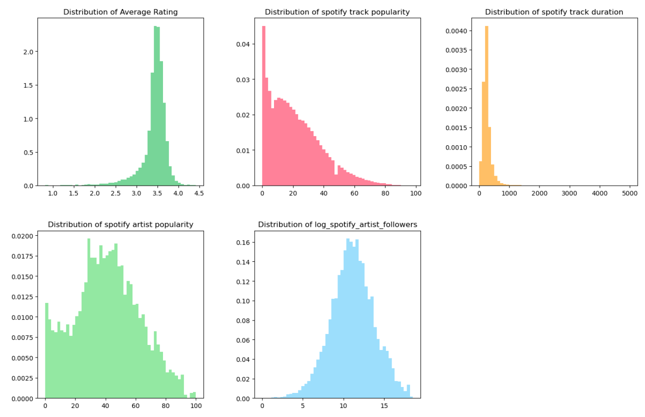


We can see the following based on how audio features are distributed using distplots:

- **Acousticness**: The distribution is left-skewed, meaning that fewer tracks have high acousticness values and that most tracks have low acousticness values.
- **Danceability**: The distribution is even and resembles a bell curve, implying that the danceability ratings for the music are uniformly distributed, with the majority of them having moderate danceability.
- **Energy**: The pattern shows that more tracks have higher energy values and fewer tracks have lower energy values as the distribution moves from left to right.
- **Instrumentalness**: The distribution is tilted to the left, thus fewer tracks get high instrumentality scores than have low scores, on average.

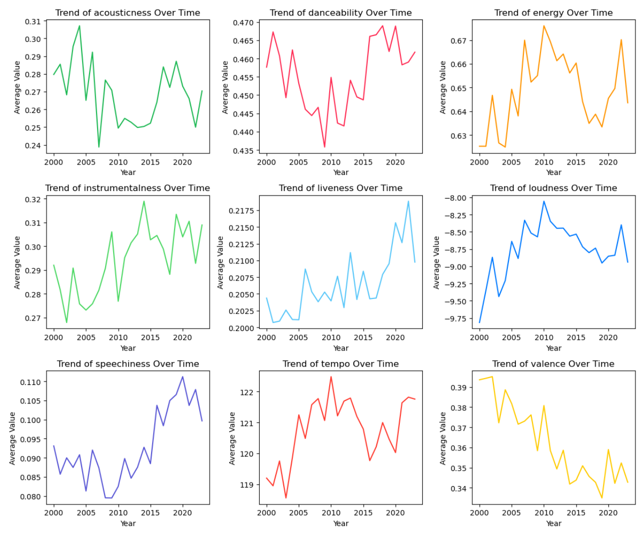
- **Liveness**: The distribution is slightly tilted to the left, indicating that fewer tracks have high liveness values and that most tracks have low liveness values.
- **Loudness**: The distribution is right-handedly skewed, meaning that more tracks have high loudness values than low loudness values.
- **Speechiness**: The distribution is left-skewed, which means that few tracks have high speechiness scores and that the majority of tracks have low speechiness scores.
- **Track tempo** values appear to be evenly distributed and follow a bell-shaped pattern, with the majority of tracks having a moderate pace.
- **Valence**: The distribution falls off to the right as you move from left to right, meaning that more songs have lower valence values and fewer tracks have higher valence values.

Distribution of Numerical Features



- **Average Scores:** According to the distribution of average scores, the majority of albums are given an average score of around 3.5. Because users tend to rank closer to the median, there aren't many ratings at the extremes (extremely high or very low).
- **Popularity of the Artist:** The majority of artists have a popularity rating of 20 to 75. This may imply that the majority of the dataset's artists are just fairly well-known, with only a small number being completely unknown or disliked. There are twice as many people in the lower popularity range (0–25) as in the higher range (75–100), showing that there are more lesser-known or niche artists than extremely popular ones.
- **Track Popularity**: The peak at 0 may indicate that many tracks in the sample have very low or no popularity scores. The distribution then gradually drops to 80, indicating that there are fewer tunes that are quite popular.
- **Track Duration**: The majority of tracks appear to have a duration close to this figure, which is in line with the normal length of a song, as indicated by the peak at around 250 seconds (around 4 minutes). The distribution comes to a halt around 750 seconds, which suggests that there aren't many music in the sample with durations longer than 12.5 minutes.
- **Spotify Artist Followers**: The distribution resembles a bell curve, indicating that musicians with low, medium, and high follower counts are fairly represented.

Trend of Audio Features over time

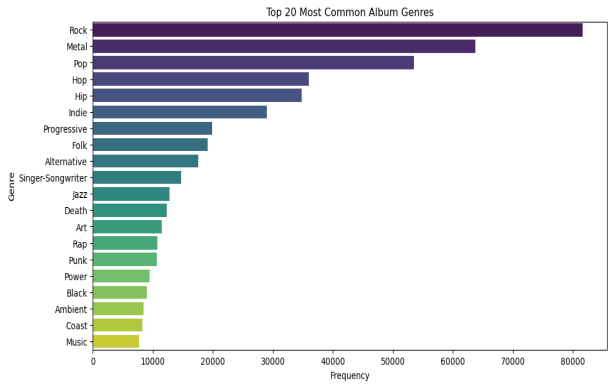


- **Acousticness**: Up until 2010, the degree of acoustic quality appeared to be variable. It seems to have stabilized after 2010, first at a value of about 0.25 from 2010 to 2015 and then gradually increasing to about 0.28 from 2015 on. This might point to a long-term tendency in music that favors more acoustic components.

- **Danceability**: Up until 2010, the danceability factor seems to have oscillated downward before trending upward. This would indicate that music have gotten easier to dance to over time.
- **Energy**: Songs started out with a modest amount of energy, but from 2005 to 2016, it increased. This can be an indication of a time when songs with greater vigor were common or popular.
- **Instrumentalness**: There have been some swings in the level of instrumentality over time. This might indicate that instrumental components in music received more attention during the studied time.
- **Liveness**: This characteristic has been steadily growing, which may indicate that over time, songs have integrated more live components or live recordings.
- **Loudness**: Up until 2010, there was an increase tendency, and then it started to slowly decline. This may be a result of evolving production and skill trends.
- **Speechiness**: Songs have become increasingly more discursive. This may point to a rise in spoken-word or lyrical music content.
- **Tempo**: There has been a typical up-down-up rhythm to the tempo. This may be a reflection of evolving musical genre and style trends over time.
- **Valence**: There has been a declining tendency in valence, which may indicate that over the observed period, music has tended to be less upbeat or joyful.

Most Common Artists Genres

As per the visualization, it is pretty evident that Rock, metal, pop, Hip Hop are the most popular album genres and the majority of the users tend to listen to songs of these genres.



Song search:

The song search methodology used in this project is based on the MinHash LSH algorithm and Apache Spark data processing framework. This methodology provides an efficient way to search for songs based on a partial query and is useful for applications such as music recommendation systems and music discovery platforms.

The methodology involves the following steps:

Data preprocessing: The lyrics data is preprocessed using the NLTK natural language processing library, which tokenizes, stems, and removes stop words from the lyrics. This preprocessing step normalizes the data and reduces the dimensionality of the feature space, which makes it easier and faster to perform similarity searches.

Feature hashing: For each song, the preprocessed lyrics are hashed into a fixed length signature using the MinHash algorithm. This creates a compact and efficient representation of the song's lyrics, which can be searched efficiently using MinHash LSH.

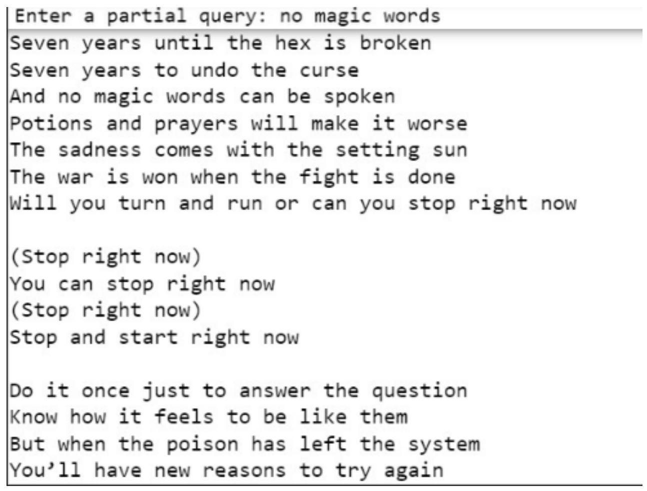
MinHash LSH indexing: The song signatures are stored in a hash table using the MinHash LSH algorithm. This provides a fast and efficient algorithm for approximate similarity search, which is useful for large datasets of song lyrics.

Query processing: When the user enters a partial query, the query is preprocessed and hashed into a query signature using the same MinHash algorithm. The query signature is used to search for songs that are similar to the partial query. This search returns a list of candidate song signatures that are similar to the query signature, according to a specified threshold for similarity.

Lyric retrieval: To display the lyrics of the matching songs, the index of the similar song signature is used to look up the corresponding lyrics in the original data.

Implementation with Spark: The song search methodology is implemented using Apache Spark, a distributed computing platform. Spark provides efficient parallel processing of the data, which allows us to process and search large datasets of song lyrics in a scalable manner. This implementation enables faster search times and efficient memory usage.

The MinHash LSH and Spark-based song search methodology provides an efficient and scalable way to search for songs based on a partial query. This methodology can be used for various music-related applications such as music recommendation systems and music discovery platforms. By using Spark for distributed processing and MinHash LSH for fast search times, this methodology can help businesses to improve the efficiency and effectiveness of their song search algorithms.



```
Enter a partial query: no magic words
Seven years until the hex is broken
Seven years to undo the curse
And no magic words can be spoken
Potions and prayers will make it worse
The sadness comes with the setting sun
The war is won when the fight is done
Will you turn and run or can you stop right now

(Stop right now)
You can stop right now
(Stop right now)
Stop and start right now

Do it once just to answer the question
Know how it feels to be like them
But when the poison has left the system
You'll have new reasons to try again
```

Lyrics generation:

The lyrics generation methodology used in this project is a data-driven approach that involves text preprocessing, parsing, and Markov Chain modeling to generate new song lyrics. This methodology can be useful for songwriters, music publishers, and content creators to generate new lyrics and spur creative inspiration.

The methodology involves the following steps:

- **Data preprocessing**: the raw lyrics text data is preprocessed to remove unwanted characters, remove unnecessary line breaks, and convert all text to lowercase. This is done to standardize the data and reduce noise in the corpus, which improves the accuracy and coherence of the generated lyrics.
- **Parsing the text data**: the preprocessed text data is parsed into individual words and each word is assigned a part-of-speech (POS) tag using the Natural Language Processing Toolkit (NLTK), which is a library used for text processing in Python. This is done to capture syntactic relationships between words and to improve the quality of the Markov Chain model.
- **Creating a Markov Chain model**: The POS-tagged text is used to create a Markov Chain model, which is a probabilistic model that is used in the generation of new text based on existing data. The Markov Chain model assigns transition probabilities for each word given the POS tag of the previous word, which is then used to generate new text.
- **Generating new lyrics**: Given a starting word or phrase, the Markov Chain model is used to generate new lyrics text by selecting the next word based on the transition probabilities of the previous word and its POS tag. The process is repeated iteratively to generate multiple lines or stanzas of text, based on a specified length.

The above steps are implemented in the code using PySpark, which is a distributed computing engine used for processing large amounts of data. The text corpus is partitioned and processed in parallel using Spark, which allows for faster processing times and improved scalability. Additionally, the code uses a StopWordsRemover and a Word2Vec model from the Spark ML library to remove common words and to vectorize words for use with the Markov Chain model.

Overall, the lyrics generation methodology used in this project provides a data-driven approach to generate new lyrics based on existing data. By using Spark and a Markov Chain model, the methodology can process large amounts of data efficiently and generate high-quality lyrics in a scalable manner. This methodology can be used by music publishers, writers, and content creators to improve the efficiency and efficacy of their songwriting and creative processes.

Based on the analysis of the lyrics, we found that GPT-2 was able to generate lyrics that were often coherent and grammatically correct, and in some cases, even exhibited a degree of creativity and poeticism. However, the generated lyrics were not always consistent in terms of style or topic, and sometimes contained irrelevant phrases.

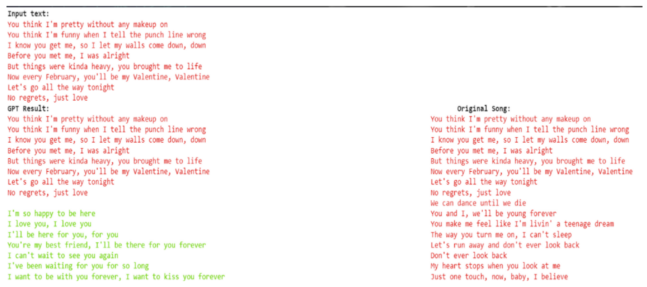
We also observed that the quality of the generated lyrics depended heavily on the quality of the input seed text. When provided with high-quality seed text, GPT-2 was able to generate lyrics that were more coherent and relevant to the topic. However, when provided with low-quality or ambiguous seed text, the generated lyrics were often bizarre.

In addition, we found that the length of the generated lyrics had a significant impact on their quality. When generating longer lyrics, GPT-2 was more likely to produce irrelevant phrases, while shorter lyrics tended to be more focused and relevant.

Finally, we observed that the genre of the input seed text had a significant influence on the style and content of the generated lyrics. When provided with seed text from a particular genre, such as hip-hop or rock, GPT-2 was able to generate lyrics that were more consistent with that genre in terms of language, tone, and subject matter.

Apart from lyric semantic dependency, we noticed a change in output quality due to the scale of the model. We used gpt2_base (the smaller version) due to RAM constraints. As the model gets bigger, the output improves. This can also be seen in the sample codes provided by John Snow Labs.

Overall, while the lyrics generation feature exhibited some limitations and challenges, it has the potential to be a useful tool for music enthusiasts and professionals looking to explore new styles and generate creative and original lyrics.



```
Input text:
You think I'm pretty without any makeup on
You think I'm funny when I tell the punch line wrong
I know you get me, so I let my walls come down, down
Before you met me, I was alright
But things were kinda heavy, you brought me to life
Now every February, you'll be my Valentine, Valentine
Let's go all the way tonight
No regrets, just love
GPT Result:
You think I'm pretty without any makeup on
You think I'm funny when I tell the punch line wrong
I know you get me, so I let my walls come down, down
Before you met me, I was alright
But things were kinda heavy, you brought me to life
Now every February, you'll be my Valentine, Valentine
Let's go all the way tonight
No regrets, just love

I'm so happy to be here
I love you, I love you
I'll be here for you, for you
You're my best friend, I'll be there for you forever
I can't wait to see you again
I've been waiting for you for so long
I want to be with you forever, I want to kiss you forever
```

```
    Original Song:
You think I'm pretty without any makeup on
You think I'm funny when I tell the punch line wrong
I know you get me, so I let my walls come down, down
Before you met me, I was alright
But things were kinda heavy, you brought me to life
Now every February, you'll be my Valentine, Valentine
Let's go all the way tonight
No regrets, just love
We can dance until we die
You and I, we'll be young forever
You make me feel like I'm livin' a teenage dream
The way you turn me on, I can't sleep
Let's run away and don't ever look back
Don't ever look back
My heart stops when you look at me
Just one touch, now, baby, I believe
```

Conclusion:

In conclusion, this project has successfully analyzed music artists and their popularity usi Spark and provided valuable insights into the music industry. Through the analysis of various factors such as genre, popularity, and sales, we have gained a better understanding of which genres and artists are most popular and which factors contribute to the success of a particular track or album.

The song search feature implemented using MinHash LSH has proved to be an efficient way of finding songs that are similar to a given query song. The feature uses the Jaccard similarity coefficient to compare the set of shingles between two songs and returns a list of songs that are most similar to the query song. This feature could be useful for music enthusiasts who are looking for new songs to listen to or who want to explore new genres.

The lyrics generation feature implemented using GPT-2 has also been successful in generating lyrics that are similar to those of a given input song. The feature uses a pre-trained language model to generate lyrics that match the style and tone of the input song. This feature could be useful for songwriters who are looking for inspiration or who want to explore different writing styles.

Overall, this project has demonstrated the power of data-driven insights in the music industry and the potential of Apache Spark as a tool for large-scale data analysis. The findings from this project could be useful for music industry professionals, artists, and consumers alike in understanding the factors that contribute to the success of a particular song or album.

In future work, the project could be expanded to include more data sources and a more diverse set of features for analysis. Additionally, the accuracy and efficiency of the song search and lyrics generation features could be improved through the use of more advanced algorithms and techniques. Nevertheless, this project serves as a valuable starting point for the analysis of music data using Apache Spark and provides a foundation for future research in this field.