**Introduction:**

In recent years, the field of natural language processing (NLP) has seen significant advancements in the use of machine learning techniques for text analysis. Text pre-processing is a crucial step in NLP, as it involves cleaning and transforming unstructured text data into a structured format that can be easily analysed by algorithms. In this report, we focus on a song dataset that contains information on various artists, song names, lyrics, and links to the songs. We aim to explore the pre-processing techniques used to clean and prepare the data for analysis, and present any insights gained from analysing the pre-processed dataset. By doing so, we hope to showcase the importance of text pre-processing in NLP, and the potential insights that can be gained from analysing structured text data.

Data Description:

The Dataset contain 4 columns: Artist, Song, Link, Text.

Graphical user interface, text, application, email

Description automatically generated

Text

Description automatically generated

The above figure shows that the dataset contains 57650 rows and 643 different artist data.

Text Pre-Processing:

Text Pre-Processing: To prepare the song dataset for analysis, we performed several pre-processing steps on the text data. First, we removed any white spaces and punctuations from the lyrics, as these characters do not provide any useful information for analysis. Next, we tokenized the lyrics by splitting each sentence into individual words and creating a list of these words.

After tokenization, we removed any stop words from the lyrics. Stop words are commonly used words in a language that do not convey any meaningful information for analysis, such as "the," "a," and "and." By removing stop words, we can focus on the more important words in the lyrics that may provide insights about the song.

We also performed stemming on the lyrics to reduce each word to its base form. For example, words like "running" and "runner" would be reduced to "run." This helps to group together words with similar meanings and reduce the overall dimensionality of the dataset.

Additionally, we performed lemmatization on the lyrics to further reduce the words to their base form, but in a more accurate way than stemming. Unlike stemming, lemmatization considers the context of the word in the sentence and its part of speech. However, we found that stemming worked well enough for our analysis, so we did not use lemmatization in this report.

Overall, by performing these pre-processing steps, we were able to clean and transform the raw text data into a structured format that is suitable for analysis using machine learning techniques.

Using Lyrics to predict the song’s genre:

Using lyrics as a feature to predict the genre of a song can be a challenging task, as lyrics are often highly subjective and may not necessarily be representative of the genre. However, it is still possible to use natural language processing techniques to extract features from the lyrics and attempt to cluster similar songs together.

One common technique for feature extraction is the CountVectorizer, which is a method that takes a text corpus and converts it into a vector of word counts. In this case, we could use the CountVectorizer to extract features from the song lyrics and create a matrix where each row represents a song, and each column represents a word in the corpus. We could then use dimensionality reduction techniques such as PCA to reduce the dimensionality of the feature space and cluster similar songs together.

However, using lyrics alone may not be enough to accurately predict the genre of a song. This is because the genre of a song is often influenced by a variety of other factors, such as the instrumentation, tempo, and overall sound of the song. To improve the accuracy of the predictions, we could consider incorporating additional features such as audio features (e.g. tempo, loudness, timbre) or metadata (e.g. artist name, album title, release date) into the analysis.

Overall, predicting the genre of a song using lyrics alone can be a challenging task, and it may be necessary to incorporate additional features to improve the accuracy of your predictions.

Connecting to Spotify API:

Connecting to the Spotify API using Python allowed us to gather additional information about the artists in this song dataset, such as their genre, total number of followers, and popularity. This information could be useful in generating a song rating, which considers factors such as sentiment, artist popularity, and artist followers.

To use the Spotify API, we likely first needed to obtain an API key, which would allow us to make authorized requests to the API. Once we had the API key, we could use Python libraries such as Spotify to interact with the API and retrieve information about the artists in the dataset.

To gather information about an artist, we could use the spotipy.client object to search for the artist by name and retrieve their Spotify ID. With the artist's ID, we could then use spotipy.client.artist() to retrieve their profile information, including their genre, number of followers, and popularity score.

Using this information, we could create a song rating that considers the sentiment of the lyrics, the popularity of the artist, and the number of followers the artist has. This rating could be calculated using a weighted average, with the weights determined by the importance of each factor. For example, if sentiment was the most important factor, it could be weighted more heavily than artist popularity or followers.

Once we had calculated the song ratings for each song in your dataset, we could use cosine similarity to find similar songs. Cosine similarity is a measure of similarity between two vectors, in this case the song ratings for each song. By calculating the cosine similarity between a given song and all other songs in the dataset, we could identify the 10 songs with the highest similarity scores and recommend them to the user.

Sentiment Analysis:

After performing text pre-processing on the lyrics in our song dataset, we conducted a sentiment analysis to determine the emotional tone of each song. We used the TextBlob and polarity sentiment analysis tool, which is a lexicon and rule-based sentiment analysis tool specifically designed for social media text.

Overall, our sentiment analysis provides insights into the emotional tone of the songs in our dataset and their relationship to other variables. This information could be useful for understanding the preferences of music listeners and for creating playlists based on emotional themes.

**Song Ratings:**

To create a song rating, we combined the sentiment score with the popularity and followers of the artist. Popularity and followers are proxies for the perceived quality of the artist, and by extension, the perceived quality of the song. This rating represents the overall desirability of the song for a given user, and it can be used to rank songs in terms of their relevance to the user's preferences.

Cosine similarity:

Cosine similarity is a measure of similarity between two vectors in a high-dimensional space. In this project, we used cosine similarity to measure the similarity between songs based on their sentiment score, artist popularity, and artist followers. This allowed us to find songs that were similar in terms of their overall sentiment and artist popularity. Specifically, we used the cosine similarity metric to calculate the distance between the feature vectors of each pair of songs, and then we ranked the songs based on their distance from the user's input song.

Recommendation System:

A recommendation system is a system that recommends items to users based on their preferences. In this project, we built a recommendation system that takes as input a song name and generates a list of the 10 most similar songs based on their sentiment score, artist popularity, and artist followers. Specifically, we calculated the cosine similarity between the feature vector of the input song and the feature vectors of all other songs in the dataset. Then, we ranked the songs based on their similarity score and returned the top 10 songs as recommendations.

**Dashboard using Power BI:**

Creating a dashboard using Power BI is a great way to gain insights into your dataset and present the findings in an easy-to-understand format. Power BI is a business analytics service that allows you to connect to a wide range of data sources, create interactive visualizations, and share your insights with others.

To create a dashboard for your song dataset, you likely first needed to import the dataset into Power BI. This could be done using a variety of data connectors, depending on the format of the data. Once the data was imported, you could use the Power BI interface to create visualizations such as charts, graphs, and tables, which would allow you to explore the data and identify trends and patterns.

Some examples of visualizations you might create for your song dataset could include a pie chart showing the distribution of genres in the dataset, a bar chart showing the number of songs by artist.

Once your dashboard was complete, you could share it with others by publishing it to the Power BI service. This would allow others to view your findings and interact with the data in meaningful ways.

Graphical user interface

Description automatically generated

A screenshot of a computer

Description automatically generated with medium confidence

**Conclusion:**

Overall, this project leverages text pre-processing techniques, sentiment analysis, and cosine similarity to build a recommendation system that takes into account the sentiment of the lyrics and the popularity of the artist to recommend similar songs to users. By combining these techniques, we have created a system that can help users discover new songs that they may enjoy based on their preferences. However, it's important to note that the effectiveness of the recommendation system will depend on the quality of the features used, the accuracy of the sentiment analysis, and the relevance of the cosine similarity metric to the task at hand. Additionally, it's important to consider other factors that may influence a user's song preference, such as genre, tempo, and personal taste.