Music Sentiment Analysis

Yobah Bertrand Yonkou | Sinchana R | Supraja Yadav
Nixon Joji | Saba Santhosh | Alvina Joanna

As part of our mandatory practical paper [BP1P2] Probability & Stochastic Process Lab in the Master's course of Big Data Analytics at St, Joseph's University; our group were asked to implement an algorithm to better understand the working and applications of the same.
We chose to implement sentiment analysis on song lyrics in order to classify the sentiment (emotion) of the song to either happy, sad or angry. The algorithm used to implement sentiment analysis was the Naive Bayes classifier. In this report we detail our process and outcome.

Problem statement

To build a model using R Programming and R Shiny that can classify music lyrics into one of three sentiment classes, namely: happy, sad and angry.

Data collection

To create a model for classifying lyrics into the aforementioned sentiments, we had to collect music data, focusing on the lyrics. Given that we were settling on building a supervised model, we needed data that was labelled. Manually collecting and labelling music lyrics is a cumbersome process, so, to make things easy, we sought no further than one of the world's largest music databases - Spotify. Believing that Spotify has a pretty good classification model, we created playlists for each of the above sentiments thus each song was labelled based on the playlist it belonged to.
Using the Spotify API, we fetched the songs from each playlist, labelling them according to their playlist (sentiment). Since Spotify's API doesn't give us access to music lyrics, we had to gather the attributes like artist name and song title that could then be used to fetch the lyrics from another platform. Thus, using the artist's name and song title, we used the Genius API to get the lyrics of each song. The collected data is stored in three different CSV files, as per their sentiment/playlist.

Data cleaning

The data collected - the lyrics, contained a lot of noise (unwanted data) that would not be required nor helpful for our model. We decided to strip this noise from our dataset. To do so, we performed the following operations:

1. Converting lyrics to lowercase:
   In this step, all letters are converted into lowercase. This brings the data to a consistent format (in terms of letters). Without this step, the machine would treat the same words differently if they have different cases for example "Run" will be considered different from "run" or "RUN" or "rUN" and so on.

2. Removing lyrics divisions:
   In the course of collecting our lyrics from Genius, we noticed that the lyrics were divided into subsections like chorus, pre-chorus, intro and so on. And some of the lyrics contained timestamps. All of these are not useful during sentiment analysis, thus this step gets rid of such sections. Since all these section dividers have a pattern, we made use of regex to remove time.

3. Correcting spellings:
   As human beings, we are bound to make errors in many aspects of our life including typing. This step runs each lyric through a function that searches for all typos and predicts the best possible replacement for that word.

4. Removing contractions:
   In the English language, a contraction is a combination of two or more words in order to shorten the form by dropping letters and replacing them with a single apostrophe. For example, the contraction for "does not" is "doesn't", and "are not" is "ain't". It is important to decompose contractions as it brings our data to a standardized format (just like in step 1).

5. Removing special characters (punctuations) and non-alphabetical characters (like numbers)
   Special characters are non-alphanumeric characters. These characters do not really add any value to the understanding of the text. In this step, we will get rid of all such characters.
   Moreover, numeric characters do not really contribute to the understanding of sentiment from text. Thus numbers will be removed from the dataset.
   Punctuations are marks that help us separate sentences and their parts and also to clarify meanings. though these characters might be important in the part of speech tagging of words, it won't be used during model building and EDA. So, we will get rid of these characters but will keep a copy of each lyric containing these characters for future use.

6. Removing extra spaces (if any)
   In this step, we are simply getting rid of any extra spaces that are within our text data for example replacing two spaces with one, a tab with one space and new line with one space and so on.

Feature extraction

The aim here was to use the cleaned set of lyrics to extract features that are relevant to our model. Here, we lemmatized our set of lyrics using part of speech tagging. Lemmatization is the process of bringing words to their root forms, however, depending on the part of speech of a word, the word can have different root forms thus combining lemmatization with part of speech tagging. More so, we removed some words like prepositions, which are not relevant to our model. This is because they do not help us determine the sentiment of a sentence. These words were termed stop words.

Model building

Before building our model, we had to convert our dataset into a form that is understandable by our model (Naive Bayes). To do so, we used the Document Term Matrix precisely Term Frequency - Inverse Document Frequency.

> Document Term Matrix: The document term matrix is a technique of structuring text data in such a away that documents will be placed in the row and the terms of each document in the rows of the data frame. The value of each term will be either zero or one representing the absence or presence of the term in a particular document, respectively. Each cell contains the weight of a term in a particular document. The weight of a term in a document can be determined using any of the following approaches;
>
> Term Frequency - Inverse Document Frequency: This approach seeks to quantify the importance of a term in a document amongst a collection of documents.

Finally, the output from the process above was fed into a naive Bayes classifier, building a model that can group lyrics into one of three sentiments, namely: happy, sad and angry.

Model deployment

After building the model, it had to be made available to the people who require tin a manner that is easy to use. In order words, the model has to be deployed. To deploy our model, we used RShiny, a user interface was developed. The user interface allows users to select an artist (from a dropdown) and any song

(from another dropdown, based on the artist selected) by that artist after which the selected song's lyrics are passed to our model and the probability of the lyrics being in each of the selected sentiment classes will be displayed in a bar chart.

Future work

This model was a very interesting project to work on and we identified the following areas in which the work could be incorporated for business utilization:

- To analyse the sentiments of songs being streamed by a user to suggest either similar songs or those of a different genre (to keep users engaged longer on the platform)
- To analyse the trend in sentiments over a period of time in order to predict what kind of song will more likely be accepted by an audience at a particular time

Acknowledgements

We'd like to thank our professor Dr. Srinivas Bhogle for his never-ending support and encouragement to us in making sure we brought the project to fruition.