SD201 Mining of Large Datasets

MUSIC GENRE CLASSIFICATION USING SONG LYRICS

Team members :
Yassine BENBIHI
Louiza AOUAOUCHE
Farah JABRI
Tristan PERROT

Supervised by :
Mrs Tiphaine VIARD

November 2022

Contents

1 Introduction

The mood of the human being varies a lot depending on the conditions in which he is. He then expresses different emotions according to his mood. These emotions will be an important factor in many of his choices as when he listens to music, he will naturally choose a type of song corresponding to his situation. Thus are born musical genres that will correspond to the moods, tastes and desires of each person. Beyond the sound and rhythm that characterize the songs by genre, the lyrics are also a good indicator of a song type. If we make our grandmothers read several songs she would probably say that the words from rap songs are too harsh. She would naturally be able to categorise the songs only with the lyrics.
Nowadays, Spotify and other platforms that offer a wide range of online music classify songs by genre, which allows, among other things, to suggest to users the music that suits them best. So we thought that the exploration of lyrics can be an interesting tool to establish this classification and we made it the subject of our project with this problematic : is it possible to predict song's genre using only lyrics?
The first part of our project was to build the database using a scrapping method with an API. After collecting the data, a pre-processing work regarding the lyrics of our dataset had to be done for cleaning the data.
Once the data was ready, we trained models with different approaches in order to find the most efficient one at detecting the genre of a random music.
To finish, we concluded our work by analysing the results obtained by the different methods that we used.

2 Construction of the dataset

2.1 Scrapping

To construct the dataset needed for the project, we opted for a scrapping method through the Genius [3], a website specialized in song lyrics. For reasons of complexity beyond the objectives of this course we have chosen songs in English only. To scrap all the data, we made a python algorithm using the LyricsGenius [4] client. We chose the music genres we wanted to study : Rap, R&B, Hip Hop, Pop, Rock, Metal, Jazz and Country. After that we ran the client to get all the lyrics from the Genius API (maximum 1000 songs per genre). Then, we exported it into a CSV file with '#' character as a separator. We could not use a simple comma because commas are used inside the lyrics.

2.2 Raw Data exploration

The dataset initially contains 6858 rows (songs) and 5 columns corresponding to **title** of the song, the **artist**, the **lyrics**, the **genre**, and the corresponding **url** on genius.

	artist	title	lyrics	genre	url
0	Eminem	Rap God	Rap God Lyrics\r\n"Look, I was gonna go easy o...	rap	https://genius.com/Eminem-rap-god-lyrics
1	Cardi B	WAP	WAP Lyrics\r\nWhores in this house\r\nThere's ...	rap	https://genius.com/Cardi-b-wap-lyrics
2	Kendrick Lamar	HUMBLE.	HUMBLE. Lyrics\r\nNobody pray for me\r\nIt bee...	rap	https://genius.com/Kendrick-lamar-humble-lyrics
3	Migos	Bad and Boujee	Bad and Boujee Lyrics\r\nYou know, young rich ...	rap	https://genius.com/Migos-bad-and-boujee-lyrics
4	Drake	God's Plan	God's Plan Lyrics\r\nAnd they wishin' and wish...	rap	https://genius.com/Drake-gods-plan-lyrics

Figure 1: Head of the dataset containing row data

2.3 Data cleaning

The resulting dataset of raw data is not fully operational for the purpose of the project. It is therefore necessary to do a cleaning.

Unnecessary information As said before, for each song in the raw dataset we have information about the title, the artist, the lyrics, the genre and an url. The first step of the cleaning was to keep only two columns, **lyrics** and **genre**. These columns will interest us the most since we aim to predict the genre with the song's lyrics.

Missing values Among the 6858, 25 songs were missing lyrics so we had to delete the corresponding rows because the lyrics column had to be defined for each instance.

Stopwords and punctuation A stopword is any of a number of very commonly used words, as a, and, in, and to, that are normally excluded by computer search engines or when compiling a concordance[6]. In our case, stopword is a common word that appears so often in our songs lyrics that it is useless to take it into account. At the beginning, we tried to use NLTK [5] default stopwords list which provides a good basis for starting. But we found out that it is too tight. Many other words were often repeated in songs of all kinds (e.g "cause", "yeah", "oh" etc). So we

decided to use another stopwords list inspired by [7] which helped us to better filter the lyrics. However, some words still appeared in all the genre occurrence lists (such as extreme abbreviations "goin'") so we had to add them to the final list. Thus, we got our own list appropriate to the subject.
Stopwords are not the only tokens that are useless for the prediction. We had to remove all types of punctuation too, which can be summarized by the following list : -\ ?.,/ #!%&;:{ }=_ ()ˆ

Lemmatization After removing stopwords and punctuation, we applied lemmatization on all the lyrics. Lemmatization in linguistics is the process of grouping together the inflected forms of a word so they can be analysed as a single item, identified by the word's lemma, or dictionary form [2]. In other words, it consists in referring each word to its common lexical entry, often called canonical form. For example, all the words in following list ['change','changes','changing','changed','changer'] have the same lemma : change.
Another method of standardisation of data is stemming which is a reducing of words to their word stem.Unlike stemming, lemmatisation attempts to select the correct lemma depending on the context. For example:

$$\begin{array}{ccccc} \text{Caring} & \longrightarrow & \text{lemmatization} & \longrightarrow & \text{care} \\ \text{Caring} & \longrightarrow & \text{stemming} & \longrightarrow & \text{car} \end{array}$$

We naturally opted for lemmatization sense we do not lose the context and the sense of the word.

2.4 Final dataset

After the first steps of cleaning (deleting missing values), we got a new dataset where all genres were more or less represented except for the country genre which is underrepresented.

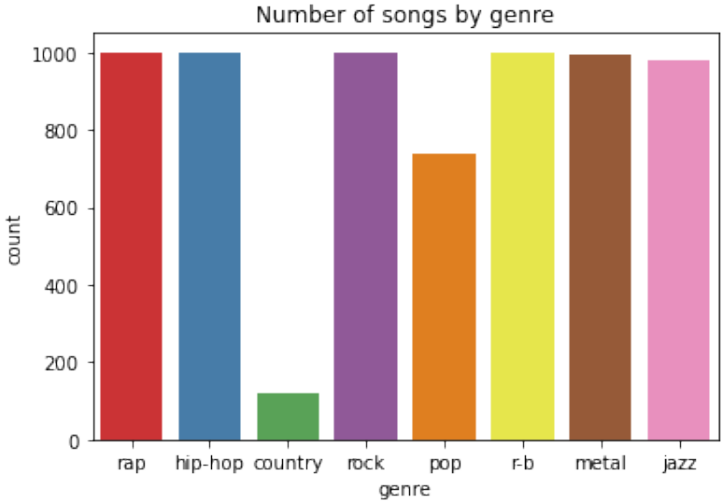


Figure 2: Number of songs by genre after removing missing values

Removing stopwords and punctuation cleaned the data and the lemmatization normalized it. Thanks to this, we obtained significant list of the most frequent words for each genre. We have made word clouds to visualize the result.

Figure 3: Word clouds by genre of music

3 Modeling

3.1 Feature extraction

As a first approach, we used CountVectorizer as a feature extractor from the lyrics. It is a function of *sklearn.feature_extraction* used to transform a given text into a vector on the basis of the frequency (count) of each word that occurs in the entire text [1]. After applying CountVectorizer, the words are not stored as strings anymore but as indexes. Let's suppose that we have two fictive songs in the lyrics column ["yeah yeah yeah girl i wanna yeah yeah yeah i wanna see you tonight yeah yeah yeah" ,"tonight is the night"].

word	yeah	girl	i	wanna	see	you	tonight	is	the	night
index	(0)	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)
lyrics 0	9	1	2	2	1	1	1	0	0	0
lyrics 1	0	0	0	0	0	0	1	1	1	1

Figure 4: Representation of CountVectorizer function

After some research, we thought that a TfidfVectorizer may be more suited in our case, as it not only take into account the number of occurrences in an instance, but also in the entire corpus. Words highly frequent in our dataset are excepted to be less relevant because they appear in several genres. [8]

word	yeah	girl	i	wanna	see	you	tonight	is	the	night
index	(0)	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)
lyrics 0	0.58	0.06	0.13	0.13	0.06	0.06	0.04	0	0	0
lyrics 1	0	0	0	0	0	0	0.17	0.27	0.27	0.27

Figure 5: Representation of the same corpus with TfidfVectorizer

3.2 Models

After transforming the data related to the lyrics we tried the accuracy of two models on predicting the music genre knowing the lyrics: Bernoulli Naive Bayes and Multinomial Naive Bayes.
The Naive Bayes algorithm is a probabilistic classifier, which relies on the very strong assumption that the elements we are studying are all independent (a limited assumption in our case, because the words of the lyrics of a music very often depend on one another), that is why it is a naive classifier.

3.2.1 Bernoulli Naive Bayes model

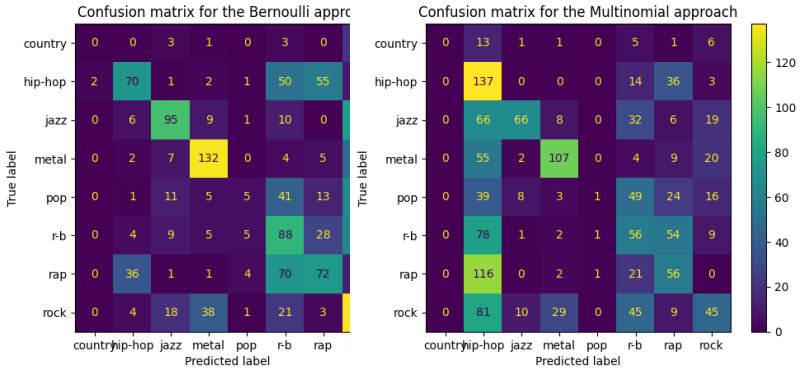
We thought about using Bernoulli Naives Bayes model as a first approach with the supposition that our lyrics could have (Bernoulli Variable = 1 , TfidfVectorizer $\neq$ 0) a word or not (Bernoulli Variable = 0 , TfidfVectorizer = 0).
With this model we looped over the best parameter (proportion between train and test) that optimises the accuracy and had, as a result, a score of 43,98% .

3.2.2 Multinomial Naive Bayes model

As we were not convinced by the first model, we opted for Multinomial Naive Bayes because it is used to solve issues involving document or text classification. The multinomial distribution describes the probability of observing counts among a number of categories, and thus multinomial naive Bayes is most appropriate for features that represent counts as it is our case.
As a result, with the same optimisation process as the previous point we had an accuracy of 39,18% .

3.2.3 Results

Furthermore, we generate two confusion matrices for both approaches.



(a) Bernouilli (b) Multinomial

Figure 6: Confusion matrices

In these confusion matrices, we can see some very interesting results. For example, rap, r-b and hip-hop are often becoming mixed up in both approaches. Moreover, the Bernoulli approach identifies much better rock and metal than the Multinomial approach, which identifies better the hip-hop style.

As related in the previous section, both of the used models were not satisfying enough. Not getting over 50% of accuracy predicted that something went wrong : the size of the data? the feature extraction? the chosen models? We asked these questions and have decided to study the issues in order to get a better score.

3.3 Other approches

3.3.1 New dataset

The results we obtained previously aren't that satisfiying. The dataset we scrapped may have too few values. Trying to fix this, we decided to choose a new dataset found on Kaggle to analyse the influence of the size of the dataset on the outcoming result. Two datasets are at our disposal : the first table makes the link between the artists and their musical genre (and other unnecessary information for this project); the second dataset gives the lyrics of each song and the name of the corresponding artist.

As for the previous dataset, we had to begin with a data-cleaning process on the raw dataset to remove unecessary information, stopwords and punctuation from the lyrics columns of the second dataset. Once this work completed, we had to merge the two tables to obtain one single table containing the lyrics of a song

and the corresponding musical genre. For practical reasons, we have grouped some categories together such as metal and heavy metal.

The dataset obtained contains 85276 lyrics which represents about ten times more values than the previous dataset. All genres are well represented with a dominant number of lyrics for the 'rock' category.

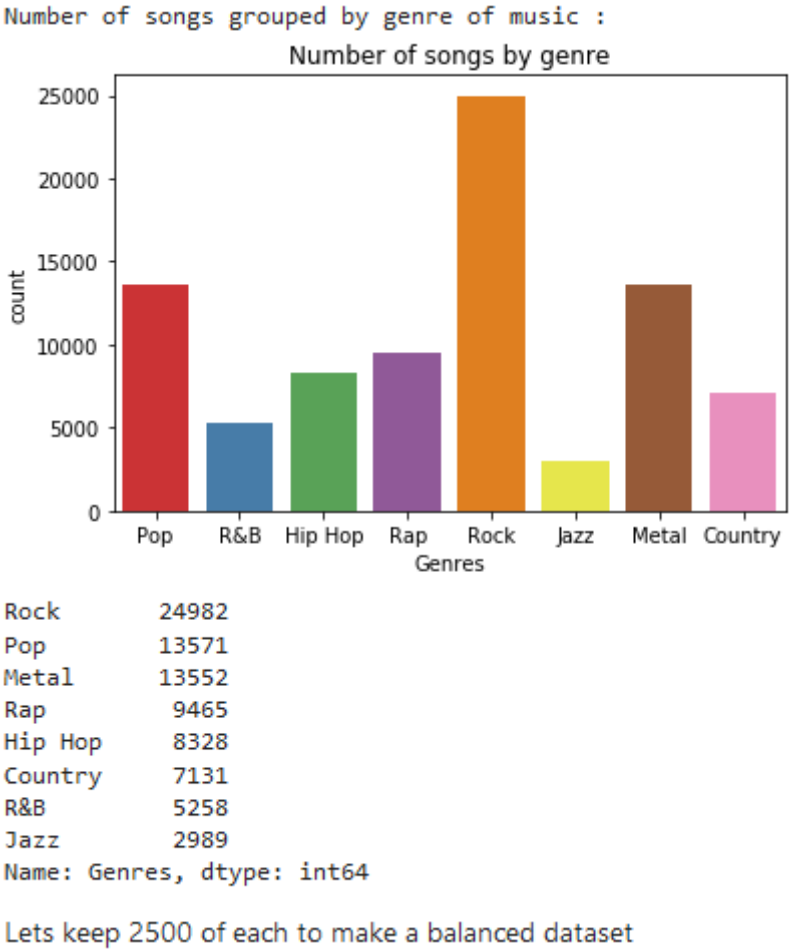


Figure 7: Number of songs by genre after removing missing values

In order to avoid mistakes due to a wrong balance between the number of lyrics for the different categories and to avoid having over or under-represented categories, we decided to set the number of songs per category to 2500. Hence, the final dataset contains 20 000 lyrics. After applying lemmatization, as explained above 2.3, on the new dataset, we obtained these word clouds for each music genres.

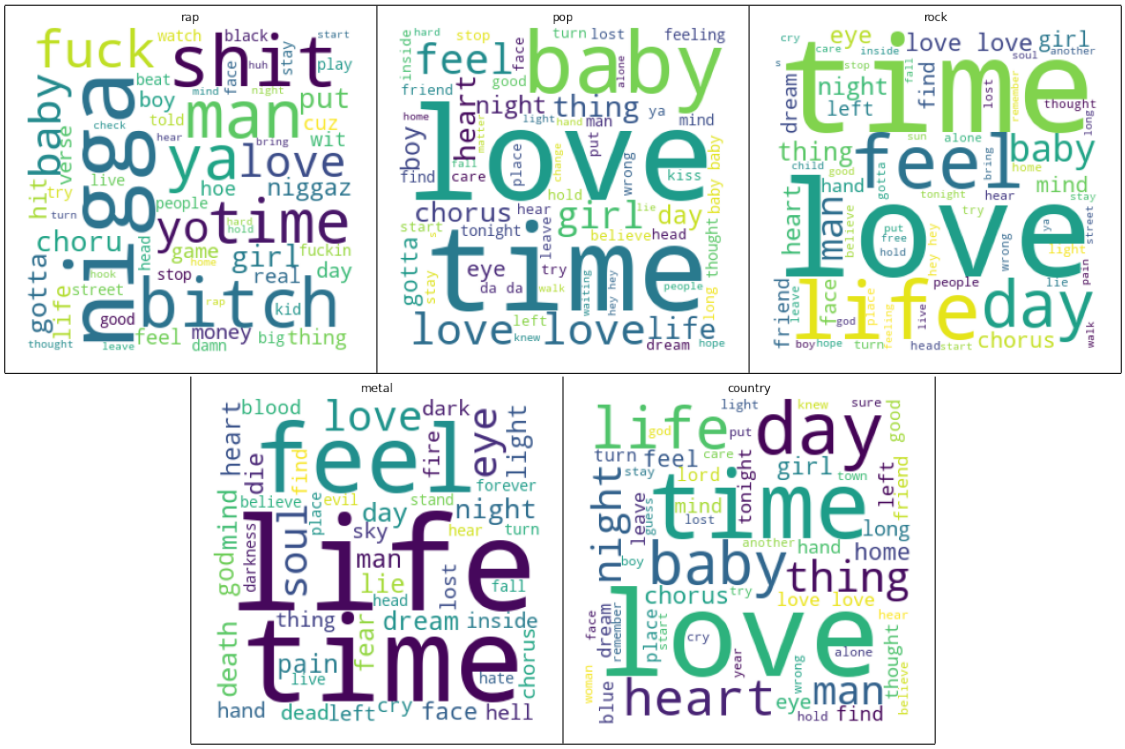


Figure 8: Example of word clouds by genre of music

The word clouds obtained by this new dataset is consistent regarding the first word clouds we had for the previous dataset (Figure ??). However, the cloud obtained for the metal type is quite different from the one found with the first dataset. Here, the hit words are more visible: "day" , "time" an "love". These hit words were also present in Figure ?? but were at the same level as the other words.

3.3.2 Models testing on the new dataset

We used the same methods used for the previous part (Binomial Naive Bayes and Multinomial Naive Bayes) on TfidfVectorizer features to compare the results and state if the size of the dataset has a huge impact on the outcome.

3.3.3 Results

After splitting the data into Train and Test we launched the models and calculated the results.

Bernoulli Naive Bayes Model With this model we obtained an accuracy of 50,52%.

Multinomial Naive Bayes Model With this model, we obtained an accuracy of 51,94%.

Comparison with the first try The result of the second experiment was more satisfactory than the one given by the first experiment; the level of accuracy was much higher for the second experiment. It means that the size of the dataset we first use was not large enough and balanced enough. Therefore, the size and the balance of the dataset have an impact on the final result.

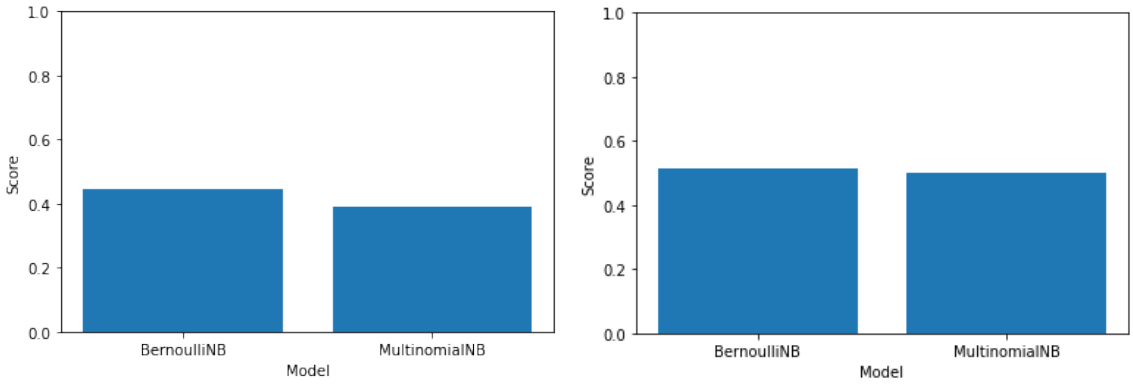


Figure 9: First dataset - Scores of models - model 1 on the left and model 2 on the right

Model	**Bernoulli NB**	**Multinomial NB**
Scrapped dataset - GENIUS	43,98%	39,18%
Larger dataset - Kaggle	50,52%	51,94%

Figure 10: Comparison of accuracy levels

We notice that, independently of the dataset size, the Bernoulli Naive Bayes model is the method giving the highest level of accuracy, the Multinomial Naive Bayes model is a little bit less efficient for these experiments.

The best level of accuracy is obtained for the second dataset using the Bernoulli Naive Bayes model.

Another thing that may have influenced the results Comparing the two datasets, we notice that the top-words for the categories 'pop', 'rock' and 'country' are very similar; words such as 'love', 'time' or 'baby' are the most used in these categories. It may also considerably influences the results.

4 Conclusion

This project was aimed at most accurately identify the genre of a song analysing the lyrics only. We first described the data cleaning process which is the pre-processing of the lyrics, the choice we made for the language, the stop-words and the normalization of the words. Then, we focused on two models to evaluate the accuracy of predicting the music genres knowing the lyrics: Bernoulli Naive Bayes model and Multinomial Naive Bayes model.

The first experiment, using our own scrapped dataset wasn't giving satisfying result; we obtained an accuracy under 45% for both of the used models. We conclude that this data might be too small adding to the fact that they were under represented categories in the data, the different categories weren't well balanced.

We decided to question the parameter corresponding to the size of the data and to check whether the data size has an influence or not on the result given by both of the methods. Moreover, we decided to balance the number of lyrics given by each category. As a result, we obtained a higher accuracy level for both of the used models. Thus, the size of the dataset is a parameter that influences the identification of a song using its lyrics.

An other parameter caught our attention : the number of categories. On the notebook we presented an annex using the same Kaggle dataset with only 5 genres (pop, rock, rap, country, metal). The results were significantly higher with 63.2% of accuracy with the Bernoulli Naive Bayes and 61.36% with Multinomial Naive Bayes.

With a maximum level of accuracy of 63.2%, it is difficult to assert that we can identify the genre of a music only by analysing the lyrics, or at least with the methods used in our work.

Member	Worked on
Tristan PERROT Louiza AOUAOUCHE	Scrapping Genius, the lyrics website, and construct our own dataset then cleaning it and testing Naive Models on it
Farah JABRI and Yassine BENBIHI	Getting a bigger dataset from Kaggle, cleaning the dataset and testing Naive Models on it

References

[1] *CountVectorizer documentation*. URL: https://www.geeksforgeeks.org/using-countvectorizer-to-extracting-features-from-text/.

[2] *Definition of lemmatization*. URL: https://en.wikipedia.org/wiki/Lemmatisation.

[3] *Genius*. URL: https://www.genius.com.

[4] *LyricsGenius : Python Client for the Genius API*. URL: https://github.com/johnwmillr/LyricsGenius.

[5] *NLTK library*. URL: https://www.nltk.org/.

[6] *Stopword definition*. URL: https://www.dictionary.com/browse/stopword.

[7] *Stopwords update list*. URL: https://gist.github.com/sebleier/554280.

[8] *TfidfVectorizer documentation*. URL: https://www.geeksforgeeks.org/understanding-tf-idf-term-frequency-inverse-document-frequency/.