Second Project IRTM

Ghadamiyan Lida, Class 507 AI 20.01.2022

Lyrics Genre Classification

Project structure

Because of the fact that I used multiple notebooks, and some of the get so much time to run, I have a folder named htmls where I put the html version of the notebooks so it is easier to see them with outputs and figures plotted (also some figures -the orange ones- are interactive). I also include the ipynb version for each of them in a folder named ipynbs. There are 6 notebooks as it follows:

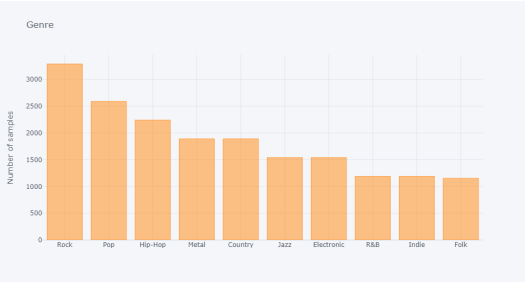
- Data visualization
- Simple Naive Bayes and unsupervised methods
- Naive Bayes without stopwords
- Bernoulli NB, SVM, KNN, MLP – here are my best results
- Bert
- SVM, RF using syllables

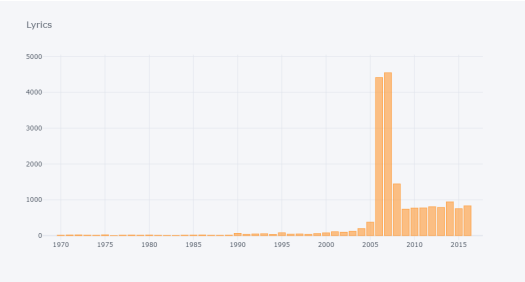
Data analysis and preprocessing

The main purpose of this project is to classify music lyrics into ten categories: Metal, Hip-Hop, Country, Jazz, Electronic, Pop, Folk, Rock, R&B and Indie.

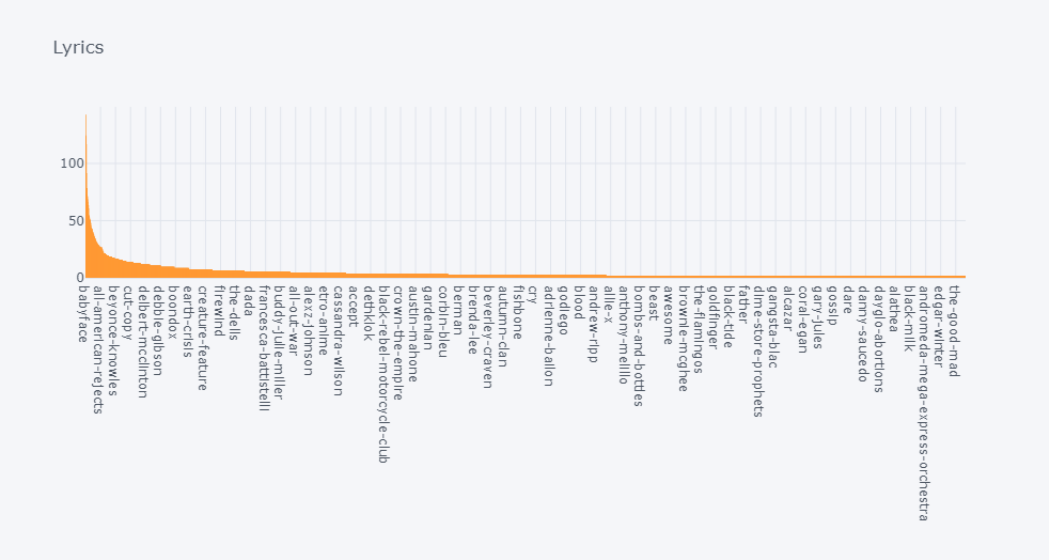
For this project we were provided two csv files, one with the training data and the other with the test data. There are 18513 entries for the train data and six features, but I will only be using the lyrics and the genre as the label. I converted the label to numbers for some of the models. Here are the distribution of the genre, year, artist and the frequency distribution of first 50 tokens.

Genre distribution Years distribution





Artists distribution



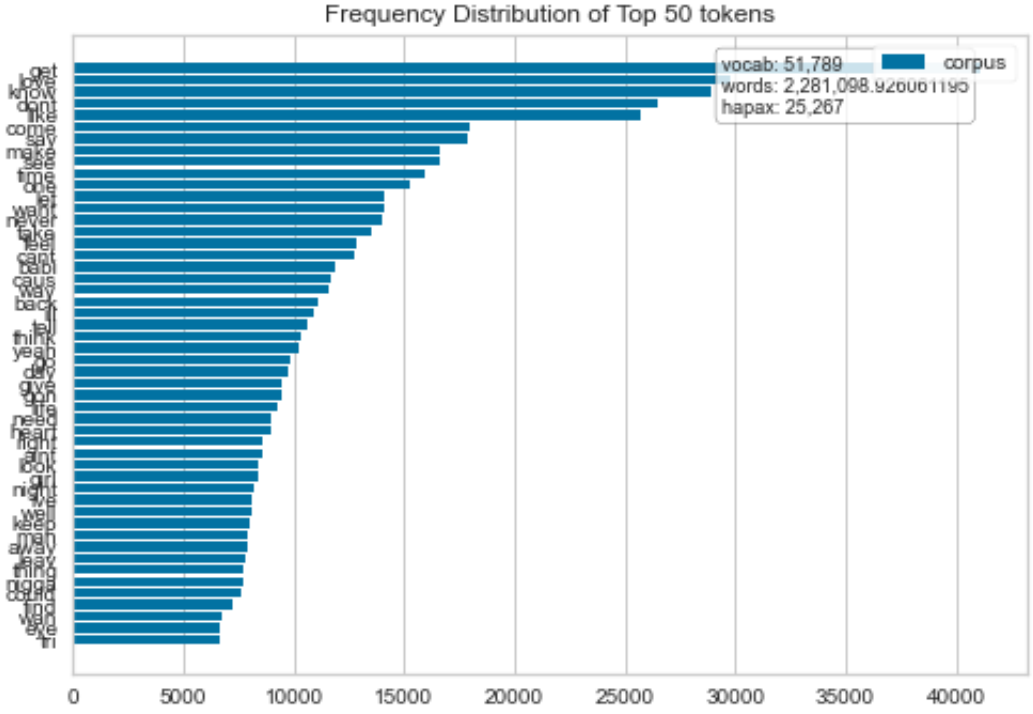
The text is in the original form, presenting uppercase letters, punctuation, numbers and '\n' at the end of each line. In order to bring the text to a cleaner point I used the nltk library for tokenizing, lemming and stemming. The punctuation was removed using string.punctuation, which is a predefined string containing punctuation symbols. After that, I removed the least two characters of every sentence, which were '\n'. I removed only words shorter than 3 letters, and numeric characters. After these steps, the tokens were put together in sentences, and the sentences were organized in a pandas data frame.

For the text tokenization I used CountVectorizer which returns for each sentence a vector where each word is described by its occurrence in all sentences. Therefore, the result for all the data is a matrix, whit a row for every sentence and a column for every word in the dictionary. The dictionary consists in a set in which every element have a key and a value, namely a word (only form the training data) and its occurrence. After this, I used TfidfTransormer to compute the frequency of each word instead of the occurrence.

In the figure below is shown the data before and after preprocessing.

	lyrics	preprocessed lyrics	Genre	Prep genre
0	I am a night in to the darkness, only soul los...	[night, the, dark, onli, soul, lose, with, wal...	Metal	0
1	Yeah\nSometimes, i just wanna fly away.\nThey ...	[yeah, sometim, just, wan, fli, away, they, sa...	Hip-Hop	1
2	Do you work hard?\nDo you work hard?\nYou don'...	[you, work, hard, you, work, hard, you, dont, ...	Metal	0
3	You know what? I'm destined to be the last man...	[you, know, what, destin, the, last, man, stan...	Hip-Hop	1
4	There ain't nothing that I would rather see\nT...	[there, aint, noth, that, would, rather, see, ...	Country	2

The distribution of 50 most common words, after the comments were preprocessed is plotted below.



Results

In order to complete this task, I started by analyzing the data. Then I tried some approaches like predefined scikit learn models, pretrained BERT and then I used the average number of syllables per verse to train the models. The best result was obtained by Bernoulli Naive Byes. I also tried training the model keeping the stop words, thinking it might be similar to the authorship task and maybe I will get better results, but it was the same. I used unsupervised learning too, as I thought maybe there are some similarities that might be found using this methods, but K-means and DBSCAN got the lowest accuracy so I did not include them in this report.

Model	Precision
NB	0.41
NB without stopwords	0.41
Bernoulli NB	0.42
MLP	0.33
SVM	0.41
KNN	0.28
BERT	0.12
SVM + syllables	0.193
Random Forest + syllables	0.194