Classification of song genres based on the lyrics

Natural Language Processing – Final report - Group 9

Amir Bachir[a], Pablo Laso[a]

[a] *University of Twente, Enschede, Netherlands*

October, 2022

Abstract

Song classification into music genres is a hard task due to the artistic nature of music which means that these classifications are often subjective and controversial, and some genres may also overlap. The researches on this topic in most cases try to extract knowledge from audio features of songs.

In this project, we look for a distinction between song lyrics which belong to different genres. We also research how accurate and reliable NLP tools can be for genre prediction. We classify songs in five genres by training a machine learning classifier on a vector representation of the songs' lyrics. We explore three different ways of embedding text and use them as input for different classifiers to compare the results.

1. Introduction

"Next song recommendation" is a relevant part of the success of music platforms as Spotify and iTunes music. The recommendation algorithms take into account multiple features related to the listening history of the user, and one of these is the song genre. Songs are usually assigned to the same genre as the one of the authors, but it is not uncommon that artists play in multiple genres, so a more effective way to assign the genre is to determine the song's specific genre.
Something that differentiates one genre from another are surely instrumental, musical, rhythmic, and audio properties of the track.
Our hypothesis is that lyrics are a strong method to identify song genres.
It is a shared opinion that most of the songs talk about the same four topics: love, friendship, statement of discontent and death. The treated topic might be the same among different genres, but there are infinite possible ways to talk about them, and we want to know if the words used within the same genres are similar or distinctive from other genres.

2. Related work

When aiming at classifying song genres, we find multiple previous attempts and work on the matter. For example, D. Bužić et al. [1] used Naive Bayes on a dataset consisting of lyrics performed by Nirvana and Metallica, 207 songs in total. Upon evaluation, they obtained very good results: precision of 0.93, recall of 0.95 and F1 -measure of 0.94, therefore lyrics classification, on a small dataset, using Naive Bayes as a classifier can be considered as successful. Still, classifying song genres based on the lyrics alone is a difficult task. That is why there have been many attempts and different approaches to this challenge. For instance, A. Tsaptsinos [2] uses a Hierarchical Attention Network to catch the hierarchical layer structure that lyrics exhibit.

3. Dataset and Data Processing

The data set was obtained from tmthyjames GitHub project repository "Cypher" [3]. Finding a significant and "correct" lyrics database is difficult due to copyright issues, in fact, this one is built scraping web pages. The data was scraped by the author through the python Cypher library. The dataset has 2,778,359 rows. The columns are 'album', 'song', 'artist, 'album_genre', 'genre', 'year', 'lyric' and 'ranker_genre'. We keep the 'ranker_genre' column as song genre label, obtained through the Ranker API, which has seven unique values ('Hip-Hop', 'Rhythm and blues', 'Pop', 'Heavy metal', 'Screamo', 'Punk rock', 'Country') and has no null values. For the purpose of our project we unify the songs belonging to the 'Heavy metal', 'Screamo', 'Punk rock' genres under the new genre 'alt rock' because of their subgenres properties. So our final genres are five: 'Hip-Hop', 'Rhythm and blues', 'Pop' and 'alt rock'. Furthermore, the lyric of each song is divided on multiple rows, so we processed the data in order to have the complete lyric in one cell. Now the dataset has 62,155 rows x 7 columns. There are two problems of unbalance in the data: first, dataset dependent, the number of songs per genre is not uniform, as shown in Figure 1; second,

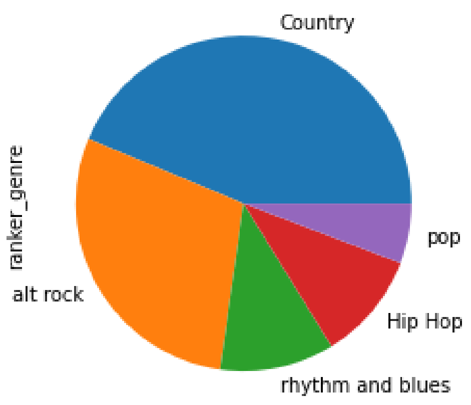


Figure 1: Song genres distribution in the dataset.

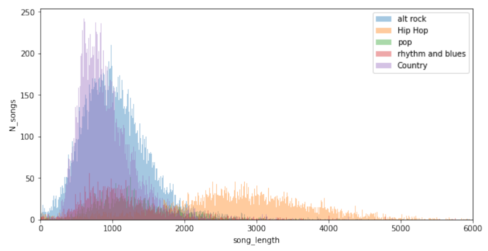


Figure 2: Lyrics length per song genre.

dataset independent, the number of words per song varies significantly between genres (Figure 2, e.g., Hip-Hop songs have on average more than the double of words per lyric with respect to other genres. The first problem is handled by sampling the same number of songs per genre while the second one, that may cause issues using CountVectorizer, is in part implicitly handled by the embedding/feature extraction algorithms such as TF-IDF.

As we can see from Figure 2 the distributions of songs length among genres are approximately Gaussians, but with very different mean and variance as shown by Table 1.

Genre	*Mean*	*Variance*
Alt Rock	1049	241724
Hip-Hop	2736	1119291
Pop	1271	289529
Rhythm&Blues	1168	359480
Country	906	148766

Table 1: Songs mean and variance per genre.

We cleaned the data from some of the noise, that includes nonsense tokens due to scraping problems and most probably incorrect or partial lyrics. Nonsense tokens were removed selecting via a regular expression only words formed by letters. Partial lyrics were avoided by sampling only songs which have more than 400 characters. Prior to any training, we looked at the 20 most recurrent words per genre, excluding stopwords. By this simple mean, we can already extract some interesting insights. All genres except "Hip-Hop" share some words as "time", "love", "heart" and "oh". Every genre except "Alt rock" has "yeah" as a very recurrent filler. On the other hand, "Hip-Hop" has some slang words which do not appear in any other genre. This might suggest a good performance in classifying "Hip-Hop" with respect to the other categories.

3.1. Training dataset

Finally, the data set we use to train our models is composed by two columns: 'lyric' and 'ranker_genre'. From lyric, we will extract our features, while 'ranker_genre' is the label we want to be able to predict. We sampled 3500 songs from each genre, with at least 400 characters per lyric. We further divided it into training (75%) and test (25%) sets, still with balanced genres.

4. Methods

We need a vector representation of our songs to feed to a classifier in order to train it. We need to determine what will be the features and the values of the vector. The better representation we have, the better the classifier will discern genres.

4.1. Count Vectors

One of the simplest embeddings is to represent a document as a bag of words (BOW). BOW means representing a document as a vector of dimension size of vocabulary, where the features are the words of the vocabulary and the values are the number of occurrences of that word in the document. Thus, we identify a document as the count of its unigrams.

It is simple and quite effective, but has also some downsides. The vector will be sparse (most of the values will be 0) because usually in a song appears a restricted subset of the vocabulary and the values depend on the length of the song: longer songs will have higher values. This could be a problem for classifiers, as KNN, if they do not use cosine similarity as a distance measure.

We used scikit learn CountVectorizer to obtain this embedding. We also removed stopwords from the vocabulary. Stop words are, in general, words which appeared very frequently in all kinds of texts, but they do not add meaning. They are used for functional purposes in sentences such as "the", "is" and "i'll". We used NLTK's stop words english set and added some stop words specific for

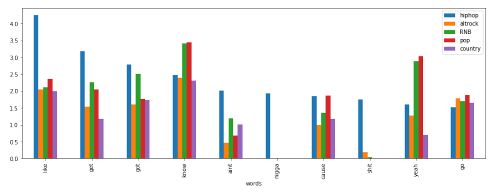


Figure 3: Hip-Hop top-10 words occurrences per genre.

our data set. For example, for all stop words like "I'll" and "would've" we added the version without apostrophe: "ill" and "wouldve", because in our lyrics we find them in that form. Finally, after some fine-tuning, we found that keeping the 2000 most important features like to the best results.

We exploited the count vectors to plot word frequency comparisons between genres. The word counts in each genre have been standardized as follows:

$$word\_count = \frac{word\_count - mean\_length_{genre}}{max\_length_{genre} - min\_length_{genre}}$$

This already brings further insights on words distributions among genres. The graph (Figure 3) shows a comparison of the min-max normalized frequency of the most relevant, in this sense, words in the "Hip Hop" category. We can see how "like", probably used in rap music to build metaphors, accounts for almost the double with respect to all other genres. Moreover, "nigga" and "shit" have almost zero relevance for other genres. This confirms the ease of identification of a "Hip Hop" song.

4.2. TF-IDF Vector

The second method of embedding a song into a vector is by using the term frequency – inverse document frequency of the lyric's words. The features are the 2000 most relevant word types and the features values are calculated as follows:

$$tf(t,d) = \frac{f_{t,d}}{\sum_{t' \in d} f_{t',d}}$$

$$idf(t) = \log[(1+n)/(1+df(t))] + 1$$

$$tf\_idf(t,d) = tf(t,d) \cdot idf(t)$$

Notice that in idf(t) we used smoothing to prevent 0 divisions in the document does not contain a word. This could happen in testing phase. In this way we scaled down the impact of tokens that awkward very frequently in our lyrics and therefore, empirically, less informative for our classification task, with respect to tokens which appear in a small subset of our data set.

We still remove stop words before computing the vector values. However, a downside of this embedding is still sparsity. For this reason, we apply dimensionality reduction using truncated singular value decomposition (SVD) which is known as latent semantic analysis (LSA) in the context of the of TF-IDF matrices. We keep the first 100 principal components. We used scikit learn TfidfVectorizer to obtain this embedding.

4.3. Word2Vec

The third and last embeddings are obtained by averaging the words vectors representation learned through Word2Vec. We explored 2 variations of this embedding:first we used the word vectors of the pre-trained Word2Vec model on the Google News data set; second, we trained the Word2Vec model on our corpus in order to embed case specific information in the words vectors. Word2Vec is used to compress the sparse representation into 300 features, as a skip-gram model that is learning to predict the word given a nearby word. Once we have the word embeddings, we represent the songs as the average of the word vectors.

The advantage of these vectors with respect to the ones of table with down vectorizer and TF-IDF vectorizer is that they are dense, not sparse, and they carry context information, because the embedding of each word depends on the surrounding words. In this way each song vector is not carrying only information about words frequencies but also words meaning expressed in function of their context. We used Word2Vec from gensim library to perform this embedding.
So the contribution to the song's representation vector will be given only by words present in the Google News dataset, in the first case. This could be a problem mainly for the "Hip Hop" songs which contain many slang words.

4.4. Model training and Evaluation

Now we have a five class-balanced and independently preprocessed training set. We feed different ML models with each differently processed dataset.

Six classifiers are fitted with the train data, namely Logistic Regression (LR), K-Nearest Neighbors (KNN), Decision Tree (DT), Multi-Layer Perceptron (MLP), Random Forests (RF), and Support Vector Machine (SVM). Naive Bayes could not handle negative values produced by some methods, for instance, SVD or Word2vec -so it was left apart in some cases. Each of the aforementioned models is fitted with train data processed by different NLP techniques, to find the most efficient one.

The performance is measured by means of the accuracy, recall, and F score. Since this is a multi-class classification problem, metric results will vary per genre and model. We compare different models by means of the test accuracy. For further insights, we dig deeper into how each classifier performs for each genre, by also considering recall and F score. In a more graphical manner, we also do so by means of the confusion matrix.

5. Results

Performance was measured by means of the accuracy, recall, precision, and F1 score. We evaluated all models and all our NLP techniques. In Figure 4 we can observe different accuracy scores for each of the aforementioned classifiers, and each resulting dataset after the feature extraction (TF-IDF, TF-IDF + SVD, Word2Vec, Word2Vec self-trained and CountVectorizer). It shows that the highest values were given for MLP, SVM, and RF classifiers. More specifically, both CountVectorizer and TF-IDF (with RF) account for the best score (average 73.25% test accuracy), closely followed by SVM (average 71.5%). Word2Vec accounts for the lowest scores. SVD does not seem to increase significantly scores -as shown when used with TF-IDF.

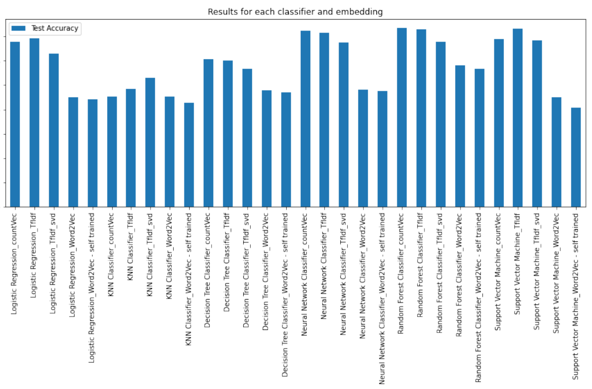


Figure 4: Test accuracy for combination of lyric embeddings and classifiers

In Table 2 we have represented the test accuracy for each classifier, trained with different datasets, namely TF-IDF, Word2Vec, and Count Vectorizer. Similarly, we can observe, in Figure 5, the Confusion Matrix of the RF classifier fitted with TF-IDF training data. The diagonal values are the correctly classified samples. The x-axis corresponds to the true label, whereas the y-axis represents the label the samples were actually assigned to by the classifier. A similar case is given for other classifiers, such as SVM (see A.11).

Further evaluation statistics performed on RF (see A.12) show high values for Hip-Hop metrics, i.e., a precision of 0.91, a recall of 0.95, and an F-score of 0.93. Contrarily, the lowest F scores are given for RNB and Country. Alt Rock accounts for the lowest precision, although F-score values are close to the average. Both Alt Rock and Pop show metrics results ranging from 0.65 to 0.72 in precision and an F-score of 0.70.

$Model$	$CountVec$	$Tf-Idf$	$Word2Vec$
Logistic Regr.	0.678	0.691	0.449
KNN	0.453	0.482	0.452
Decision Tree	0.605	0.600	0.480
MLP (NN)	0.723	0.713	0.481
Random Forest	**0.735**	**0.730**	0.580
SVM	0.689	**0.732**	0.450

Table 2: Test accuracy results for ML classifiers on CountVec, Tf-Idf, Word2Vec processed data.

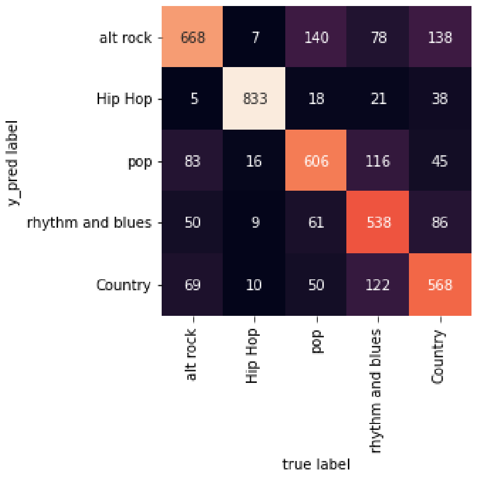


Figure 5: RF on TF-IDF vectors Confusion Matrix.

6. Discussion

The highest results are given by LR, MLP, RF and SVM in the TD-IDF and Count Vectorizer train set. However, SVM and especially RF reach maximal performance among all. Accuracies such as that of DT or KNN are very low, which makes these models unreliable. Therefore, we can state that TF-IDF and Count Vectorizer for processing together with RF for classifying is a suitable technique for our goal.

Based on Figure 5, we noticed that Hip-Hop accounts for the least errors. In Figure 3, we can actually observe that the words for Hip-Hop appear with a distinct frequency than other genres, which makes it differentiable and easier to classify. Contrarily, “alt Rock” and ”Country” have a higher number of common samples. In fact, even the top-10 normalized word counts for Country (Figure A.10), which tend to be more characteristic of each genre, are actually similar for the rest of genres. For example, "love" occurrences are very similar to those of "pop", and it is a popular word in all other genres, too. It is a similar case for "know". Actually, the following word counts are more and more similar to other genres. In other words, Country vocabulary does not show such a high differentiating potential as that of Hip-Hop. Other genres fall in between these two, although none of these get close to that of Hip-Hop.

Furthermore, RF statistics (A.12) suggest that Hip-Hop accounts for the highest scores in all metrics (precision, recall, and F-score), all of them higher than 90%. This score was achieved just by means of NLP processing techniques, which proves these methods can be very useful when it comes to genre prediction. However, the statistics also show that other genres are not so accurate (most range around the 0.65-0.70 values in the aforementioned evaluation metrics).

This might suggest that, although NLP can be very useful for some genres, it might not be enough for all song genres classification. That is, our methods could be used as a *fundamental* component of a more complex algorithm that takes into account additional data types (other than text and NLP techniques) for song genre classification.

Regarding other NLP techniques, none have shown greater potential than TF-IDF and Count Vectorizer. We believe that pretrained Word2Vec embeddings, for example, may have given worse results with respect to TF-IDF encoding because the Word2Vec vectors do not include slang words or filler words, so their contribution is not considered. With a bit of surprise, the accuracy does not increase either with the Word2Vec embeddings learned on our corpus. This might be due to the insufficient training and quality of the data.

7. Conclusion

We have tried different NLP processing techniques to treat our data. We also trained different ML models on that data and evaluated them with different metrics.

The best results are obtained when using TF-IDF and Count Vectorizer on song lyrics and utilizing RF for song genre prediction, with an accuracy of 73% for a five-class classification problem.

The easiest genre to classify is Hip-Hop, mostly due to its idiosyncratic vocabulary. Other genres, albeit still rather accurate, show some bias to other classes -such as Country for RNB and alt rock, and vice versa, which has proven to be the most difficult genre to classify.

In conclusion, applying NLP techniques on song lyrics for song genre classification is genuinely effective on some genres. It can therefore be a very powerful tool for genre classification algorithms, which might benefit from including additional data types (such as rhythm or audio) to make up for the complexity that certain genres pose.

References

[1] Bužić and J. Dobša, *Lyrics classification using Naive Bayes* (41st International Convention on Information, Communication Technology, Electronics, and Microelectronics (MIPRO), 2018, pp. 1011-1015, doi: 10.23919/MIPRO.2018.8400185, 2018).

[2] Bužić and J. Dobša, *Lyrics-based music genre classification using a hierarchical attention network* (arXiv preprint arXiv:1707.04678, 2017).

[3] tmthyjames, *Cypher*, GitHub, (2017) https://github.com/tmthyjames/cypher.git (visited on 10/18/2022).

Appendix A. Additional Information

We show below other figures and graphs that might be useful to gain a better insight of the dataset, but were omitted for the sake of simplicity and conciseness.

Appendix A.1. Top words in the whole dataset

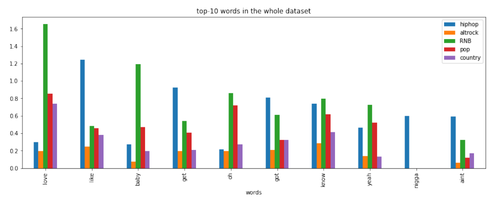


Figure A.6: Top-10 dataset words per genre.

Appendix A.2. Top words per genre

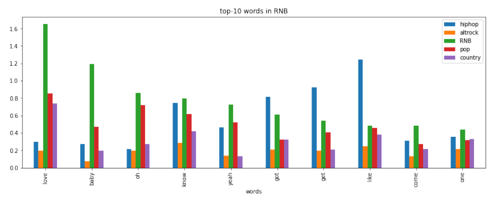


Figure A.7: Top-10 RNB words per genre.

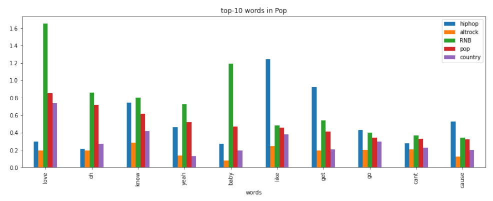


Figure A.8: Top-10 Pop words per genre.

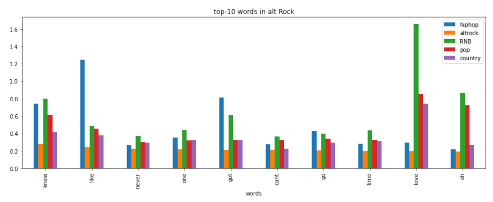


Figure A.9: Top-10 alt Rock words per genre.

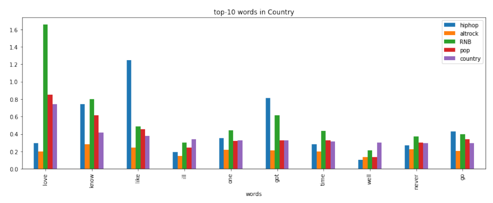


Figure A.10: Top-10 Country words per genre.

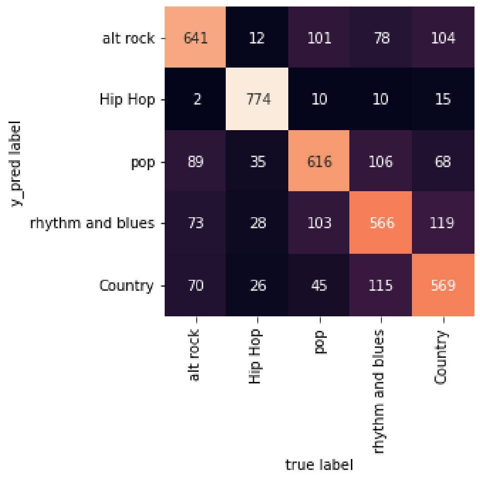
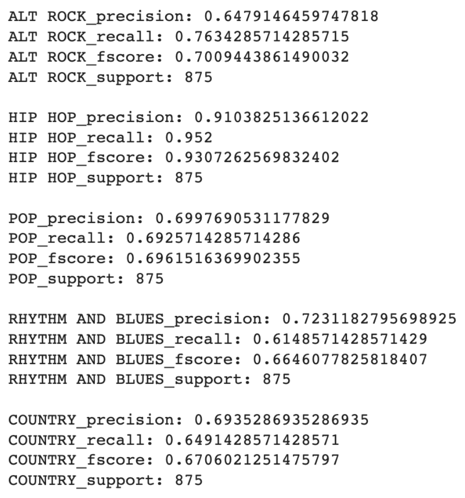


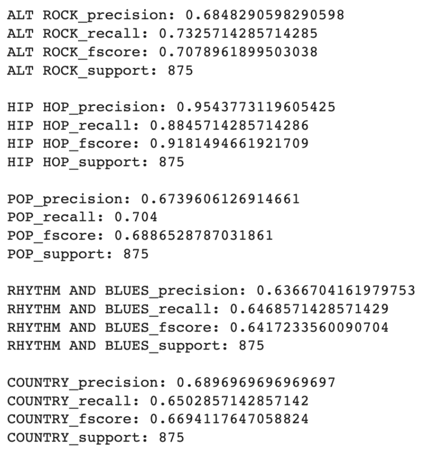
Figure A.11: SVM Confusion Matrix.

Appendix A.3. Evaluation



```
ALT ROCK_precision: 0.6479146459747818
ALT ROCK_recall: 0.7634285714285715
ALT ROCK_fscore: 0.7009443861490032
ALT ROCK_support: 875

HIP HOP_precision: 0.9103825136612022
HIP HOP_recall: 0.952
HIP HOP_fscore: 0.9307262569832402
HIP HOP_support: 875

POP_precision: 0.6997690531177829
POP_recall: 0.6925714285714286
POP_fscore: 0.6961516369902355
POP_support: 875

RHYTHM AND BLUES_precision: 0.7231182795698925
RHYTHM AND BLUES_recall: 0.6148571428571429
RHYTHM AND BLUES_fscore: 0.6646077825818407
RHYTHM AND BLUES_support: 875

COUNTRY_precision: 0.6935286935286935
COUNTRY_recall: 0.6491428571428571
COUNTRY_fscore: 0.6706021251475797
COUNTRY_support: 875
```

Figure A.12: RF Evaluation Statistics.



```
ALT ROCK_precision: 0.6848290598290598
ALT ROCK_recall: 0.7325714285714285
ALT ROCK_fscore: 0.7078961899503038
ALT ROCK_support: 875

HIP HOP_precision: 0.9543773119605425
HIP HOP_recall: 0.8845714285714286
HIP HOP_fscore: 0.9181494661921709
HIP HOP_support: 875

POP_precision: 0.6739606126914661
POP_recall: 0.704
POP_fscore: 0.6886528787031861
POP_support: 875

RHYTHM AND BLUES_precision: 0.6366704161979753
RHYTHM AND BLUES_recall: 0.6468571428571429
RHYTHM AND BLUES_fscore: 0.6417233560090704
RHYTHM AND BLUES_support: 875

COUNTRY_precision: 0.6896969696969697
COUNTRY_recall: 0.6502857142857142
COUNTRY_fscore: 0.6694117647058824
COUNTRY_support: 875
```

Figure A.13: SVM Evaluation Statistics.