Classifying Rap Subgenres by Song Lyrics

Nikita Andreikin
nandreik@gmu.edu

Table of Contents

Introduction 2
Related Work 2

Approach 2

Experiments 3
- Data 3
- Setup 3
- Results 4
- Conclusion 8

Code 9

References 9

Introduction

The goal of this project was to answer a question: can rap lyrics be used to classify a subgenre of rap? There exist a wide range of genres today that classify different types of music. There are even more subgenres that exist within them which further separate music of the same main genre. In this project, the main genre is rap and ten subgenres of rap were chosen to classify a song: trap, r&b, alternative, drill, east coast, west coast, gangsta rap, uk, cloud rap, and atlanta. Three different natural language processing classifiers were used in this project: K-nearest neighbor (KNN), decision tree, and neural network. These algorithms were trained on the lyrics of songs from the ten chosen subgenres and then tested to classify the subgenres of new rap songs. Accuracy, precision, recall, and F1-score were the metrics used to give a meaningful evaluation of the results.

Related Work

The idea of classifying a song's genre by lyrics is not new and has been attempted before. There have been many experiments done in this regard, however, they focussed on classifying songs from different main genres. Examples of this are projects by Yang (2) and Canicatti (3) which categorized a song's lyrics to one of several different main genres. Yang's approach combined a bag of words and parts of speech analysis to jointly classify a song's genre, with results ranging from 50% to 65% accurate depending on the algorithm used. Canicatti also used a bag of words approach to classify songs with results ranging from 30% to 40% accuracy depending on the algorithm. Another example is a project by Fell and Sporleder (4) which tackles this problem in more depth. Fell and Sporleder also utilized a bag of words approach to classification like previous examples, but also developed a model that analyzed the complex semantics and style of a song to achieve an average of 52.5% accuracy.

The examples of the related work previously mentioned are very similar to this project in that they utilize a bag of words to capture the word frequency of songs from different genres. The idea is that different genres use some words more or less frequently, which would give an indication of the genre a song belongs in. This project differs from these examples in that the goal is to classify between subgenres of rap, not between main genres such as rock and country.

Approach

The initial step to this problem was first finding a sufficient data set of rap song lyrics (1). Next, the data was reduced to only the ten subgenres that were chosen for this project, this step is explained in the "Setup" of the Experiments section below. The algorithms that were used to classify the songs were KNN, decision tree, and neural network.

The KNN algorithm was written and implemented by the author. The algorithm takes a list of train vectors, train subgenres, test vectors, and a "K" integer variable for the number of neighbors as input parameters. For every test vector, the algorithm looks through all train vectors and finds K nearest neighbors by computing the cosine similarity between the test vector and training vectors. Once K neighbors have been found, each neighbor's subgenres are counted. After each subgenre is counted, the

algorithm picks the most frequent subgenre to label the test vector. This can create the problem of a tie with low values of K. For example, if three out of five nearest neighbors are classified as trap and east coast, there is a tie between the two subgenres. In this case, the algorithm picks the first subgenre, trap, to label the test vector. The chance of a tie occuring can be reduced by using a larger K value.

The decision tree and neural network algorithms were used from Sklearn's tree and neural network packages. Modifications to the decision tree parameters were made to make the algorithm function as an entropy tree instead of the default gini tree which gave better results. The neural network algorithm parameters were also modified to use the "logistic" activation function instead of the default "relu" and to use the "sgd" solver instead of the default "adam" to also give better results.

Experiments

Data

The first step of this project was choosing a data set and preprocessing the data. This is perhaps the most important step because the data that is chosen and how it is modified has the most influence over the results of classifying song subgenres. The original data set (1) that was used consisted of over 250,000 different songs of various genres with features such as "Artist", "Song Title", "Lyrics", "Producer", "Genre", and "Subgenre". For the purpose of this project the features that were used were artist, lyrics, genre, and subgenre. After reducing this data set to include only rap songs in the ten subgenres that were chosen, the total number of songs totalled to about 5,800 from over 300 different artists. These subgenres were chosen because they had a sizable amount of songs in the data set and were distinct from each other in style.

The number of songs in a specific subgenre varied, with some being much more popular. For example, trap and r&b had 2,300 and 1,500 songs under their labels respectively, which created a large skew in the results towards those two subgenres. To try to create a more fair comparison of the results, the experiments were run on three different variations of the same data. The first was the baseline or unchanged data which consisted only of raw word counts. The second data group was using TFIDF to give weights to the word vectors. The third group was undersampling songs of popular subgenres and oversampling the less popular subgenres to create an equal representation between the ten chosen subgenres.

Setup

To prepare the data set to be used by the classifiers, several steps were taken to clean and format the data. First, the data set that was chosen held songs from many main genres, so it was reduced to only hold songs whose main genre was rap/hip-hop. Second, the data set was reduced to hold only songs in the ten subgenres that were selected. Third, the subgenres of songs that correspond to a specific artist were reclassified as the most popular subgenre of that artist. For example, if an artist has a majority of their songs labeled as "west coast" and one or two labeled something else, then all songs for that artist are updated to "west coast". This was done to give each song an accurate single subgenre label because many songs had multiple subgenres that could throw off the results. It was also possible to instead label each song by the

most popular subgenre of that song's album. This created a problem for songs which did not belong to an album because it is not possible to find the most popular genre of only one song. While there are always exceptions, most rap artists were found to tend to stay within their subgenre which made classifying their songs under one label fairly accurate. Finally, the lyrics of each song were cleaned to not contain labelled song sections like "[Verse]" and then processed by removing stop words, stemming, and lemmatization.

Word Frequencies to Number of Features

WF	None	2	4	6	8	10
Num. Features	27,555	17,126	11,912	9,699	8,346	7,398

The next step involved converting the song lyrics into vectors of word counts. To do this a bag of words was created from all song lyrics and the frequency of each word was recorded. Then each song was converted into a vector of word counts with the same number of features as the bag of words. For a word to be considered in the vector, it was decided that it must have a default minimum length of three characters universally. Another constraint was the minimum frequency a word must have in a song to be considered. Several minimum word frequencies (WF) were tried in the experiments, ranging from two to ten. The table above showcases the feature count of a word vector for each of the tested WF values.

This step was also responsible for converting the original word vectors into weighted TFIDF vectors or sampling the vectors by subgenre if needed. The TFIDF vectors were computed with the following formula: TFIDF(Word) = (Word Count/Total Word Count) * log(Total Song Count/Count of Songs with Word). To sample the songs, song vectors and their corresponding sub genre labels were randomly chosen until each sub genre had 580 songs total (5800 songs divided by 10 subgenres).

Results

Once preprocessing and vectorizing the data was completed, the newly created data was split into four folds. Three of the folds were used to train, while the remaining fold was used to test on. This approach to training and testing the data was chosen because it was the simplest to test and cross-validate using the entire data set. To measure the correctness of the classifications four evaluation metrics were used: accuracy, precision, recall, and F1-score. These metrics were calculated from a multi-class confusion matrix with the help of Sklearn's metrics package.

The experiments were conducted on three data groups: unchanged, TFIDF, and sampled. Various WF for all three algorithms and K values specifically for the KNN algorithm were tested in the experiments to show the trend between the evaluation metrics and different features. It is very important to note that the initial experiments were all tested against the 4th of the four folds of data. The features that gave the best results for the 4th fold were tested again in the remaining three folds for cross validation and the average scores of all four folds was calculated.

Table 1: Unchanged Data Results

		Knn				**Tree**				**NN**			
WF	K	Acc	Pre	Rec	F1	Acc	Pre	Rec	F1	Acc	Pre	Rec	F1
2	3	.36	.56	.43	.36	**.43**	**.36**	**.53**	**.42**	**.48**	**.41**	**.51**	**.46**
2	10	.41	**.59**	**.46**	**.39**								
2	20	**.42**	.36	.45	.39								
4	3	.33	.41	.39	.32	.42	.34	.51	.41	.46	.38	.51	.43
4	10	.36	.31	.4	.34								
4	20	.38	.33	.4	.36								
10	3	.28	.24	.35	.27	.37	.32	.49	.33	.37	.29	.47	.35
10	10	.27	.25	.34	.28								
10	20	.3	.28	.38	.29								

*Note: Numbers are bolded to represent the best result in that column.

The above table shows the results of the experiments on unchanged data for the 4th fold. The bolded numbers in the table show the best result for that evaluation metric for each algorithm. All three algorithms performed best on the data with the lowest WF and the KNN algorithm had highest results with a K value of 10 and 20. Each algorithm managed to classify at least 39% of songs correctly according to the F1-score. The neural network stands out with a slightly higher than average accuracy and F1-score while the KNN and decision tree algorithms held the best precision and recall respectively.

Table 2: TFIDF Data Results

		Knn				**Tree**				**NN**			
WF	K	Acc	Pre	Rec	F1	Acc	Pre	Rec	F1	Acc	Pre	Rec	F1
2	3	.39	.55	.48	.4	**.45**	**.39**	**.53**	**.44**	.33	.15	.44	.22
2	10	.44	**.6**	.54	.43								
2	20	**.47**	.37	**.55**	**.44**								
4	3	.36	.46	.41	.35	.42	.35	.53	.42	.33	.15	.44	.22
4	10	.4	.34	.47	.38								
4	20	.41	.34	.49	.39								
10	3	.28	.23	.35	.27	.37	.3	.48	.33	.33	.15	.44	.22
10	10	.24	.25	.3	.26								
10	20	.27	.28	.34	.28								

*Note: Numbers are bolded to represent the best result in that column.

Table 2 shows the results of the TFIDF data group for the 4th fold, with scores similar to table 1. The best results are again achieved with the lowest WF of 2. The KNN algorithm managed to have the best results over all with a K value of 20, with the decision tree scoring similarly. One thing to note is that the neural network performed the worst out of the algorithms for data weighted by TFIDF. In this fold it can also be seen that a change in WF value had no effect on the results of the neural network.

Table 3: Sampled Data Results

		Knn				**Tree**				**NN**			
WF	K	Acc	Pre	Rec	F1	Acc	Pre	Rec	F1	Acc	Pre	Rec	F1
2	3	**.98**	**1**	**.98**	**.99**	**.28**	**.8**	**.36**	**.47**	**.68**	**1**	**.76**	**.86**
2	10	.81	.9	.83	.86								
2	20	.61	.76	.62	.63								
4	3	**.98**	**1**	**.98**	**.99**	.26	.4	.32	.36	.57	1	.66	.79
4	10	.74	.76	.89	.82								
4	20	.55	.69	.72	.71								
10	3	.58	1	.58	.73	.03	.1	.04	.06	.3	.87	.35	.49
10	10	.49	.88	.52	.65								
10	20	.2	.69	.37	.47								

*Note: Numbers are bolded to represent the best result in that column.

Table 3 shows the results of sampled data for the 4th fold, which is the most varied of the three data groups. The KNN and neural network algorithms show deceptively high results across all metrics. This can be attributed to a skewed number of subgenres in this data fold, with a majority of the labels belonging to one to two subgenres, trap and r&b. As the K value increased for the KNN algorithm it can be seen that the scores for the four metrics steadily dropped, this is likely because with low K values a “tie” occurs more often in labeling a song which created unrealistically high results. The way this tie may occur is explained in the “Approach” section above, where the KNN algorithm is described.

Table 4: Average of Results Across all 4 Folds

Data	WF	Alg	K	Acc	Pre	Rec	F1
Unchanged	2	**Knn**	20	.35	.42	.4	.35
	2	**Tree**		.43	.43	.52	.38
	2	**NN**		.49	.55	.55	.51
TFIDF	2	**Knn**	20	.38	.44	.47	.39
	2	**Tree**		.43	.44	.52	.4
	2	**NN**		.35	.2	.48	.27
Sampled	2	**Knn**	20	.36	.5	.41	.39
	2	**Tree**		.28	.51	.28	.34
	2	**NN**		**.57**	**.71**	**.63**	**.64**

According to tables 1-3 the most successful features included a WF of 2 and a K value of 20 for the KNN algorithm. These two features were used to run tests on the remaining three folds of data to give averages of scores between all four folds. Table 4 shows the average of the scores from all four folds of data for the three different data groups using these features. The scores of the KNN and decision tree algorithms remained fairly close when tested with all three data groups, with F1-scores ranging from 35% to 40%. The neural network performed the worst when trained on TFIDF data with the lowest F1-score of 27%. However, when trained on sampled and unchanged data, the neural network received the highest benefit and achieved the best scores across the board for both data groups.

Conclusion

Through numerous experiments, it can be concluded that identifying the subgenre of rap song lyrics is not highly successful. It was found that as the word frequency limit is increased, all evaluation metric scores decreased. This is likely because a higher word frequency limit allows less total words to be used in identifying individual songs and leads to a lesser probability of identifying that song's subgenre correctly. It was also found that using different measures to create more equal and fair training data such as TFIDF weighting and sampling did not substantially improve the results, with the exception of the neural network which dramatically improved when trained on sampled data. For future experiments in classifying subgenres it would benefit in reducing the number of subgenres to classify with, from ten to five. It would also benefit in finding a more fair dataset which has a more equal distribution of songs for each subgenre, without one or two subgenres being the most popular.

Code

For this project several different procedures were written and implemented by the author. First, a procedure to clean the data was created. This included stripping away unneeded text in lyrics and formatting the data to prepare it to be processed. Next, a procedure for processing the data was implemented. This step included stemming, removing stop words, lemmatizing, creating a bag of words, and vectorizing the songs. The option to sample subgenres and to calculate the TFIDF weights of words in vectors was also implemented in this step. A KNN algorithm was also implemented from scratch, the explanation of how it functions is explained in the "Approach" section above. Finally, a procedure for splitting the processed data set into 4 folds was written as well as methods for writing the results to a file and saving the scores of the confusion matrix. The decision tree, neural network, confusion matrix, and classification evaluations were implemented by their respective Sklearn packages.

References

1) Detkov, Nikita. "250,000 Lyrics over 2k Singers." *Kaggle*, 26 Nov. 2019, www.kaggle.com/detkov/lyrics-dataset#songs_dataset.csv.
2) Yang, Junru. "Lyric-Based Music Genre Classification." 2014, https://dspace.library.uvic.ca/bitstream/handle/1828/9378/Yang_Junru_MSc_2018.pdf?sequence=1&isAllowed=y.
3) Canicatti, Anthony. "Song Genre Classification via Lyric Text Mining." 2016, http://worldcomp-proceedings.com/proc/p2016/DMI8052.pdf.
4) Fell, Michael, and Caroline Sporleder. "Lyrics-Based Analysis and Classification of Music." *ACL Anthology*, www.aclweb.org/anthology/C14-1059/.