TCSS 554A: Information Retrieval and Web Search
Classification of Song Genre

Asmita Singla
uwtasmi@uw.edu

Jyoti Shankar
jyotis@uw.edu

Surbhi Goyal
sgoyal3@uw.edu

Bharti Bansinge
bhartib@uw.edu

March 19, 2020

Abstract

With the increasing magnitude of song collection available per user on the various platforms, there is a need for automating the process of structuring and classifying that collection. In this project, we have implemented the Information Retrieval and Natural Language Processing techniques to classify the genre of songs using the text-based classifier on the text-based data in the dataset, which includes lyrics text and artist names of the songs. We have performed the analysis for classification on three Machine learning models- Naive Bayes, Random Forest, and Convolutional Neural Networks.

1. Introduction

The music libraries available on various platforms like Spotify, SoundCloud, Apple Music, and so forth are getting larger by the day, and so are the number of queries fired to retrieve the music content by their users. There are various techniques implemented to structure and retrieve the musical content based on both the audio signals and lyrical content. In this project, we aim to cover a part of the problem, which is the genre classification. We have classified the genre of a given song based on the text of the lyrics by implementing Natural Language Processing (NLP) techniques and Information Retrieval (IR) techniques. In this implementation, we aimed to learn and explore how we can make the data more relevant for machine learning (ML) classification models. We have implemented the following techniques in our project:

1. **Information retrieval:** We implemented text pre-processing using stop word removal, stemming and tokenization. Feature extraction was implemented using TF-IDF vectorizer.
2. **Classification:** Our experiment includes the classifications using both Machine Learning and Deep learning models:
    a. Naive Bayes- Is known as a state-of-the-art model for classification using text.
    b. Random Forests- Is used to experiment with a tree based approach to utilize the benefit of the Tf-IDF feature vector.
    c. Convolutional Neural Network- Is used to analyse the results of various convolutional layers for text classification.
3. **Evaluation Measures:** To measure the performance of the experiment, we implemented three evaluation techniques- 10 Fold cross-validation, F-score, and confusion matrix for Naive Bayes and Random Forest. To evaluate the CNN model, we plotted the accuracy and loss curves for testing and validation scores along with the confusion matrix and F-score.

2. Related Work

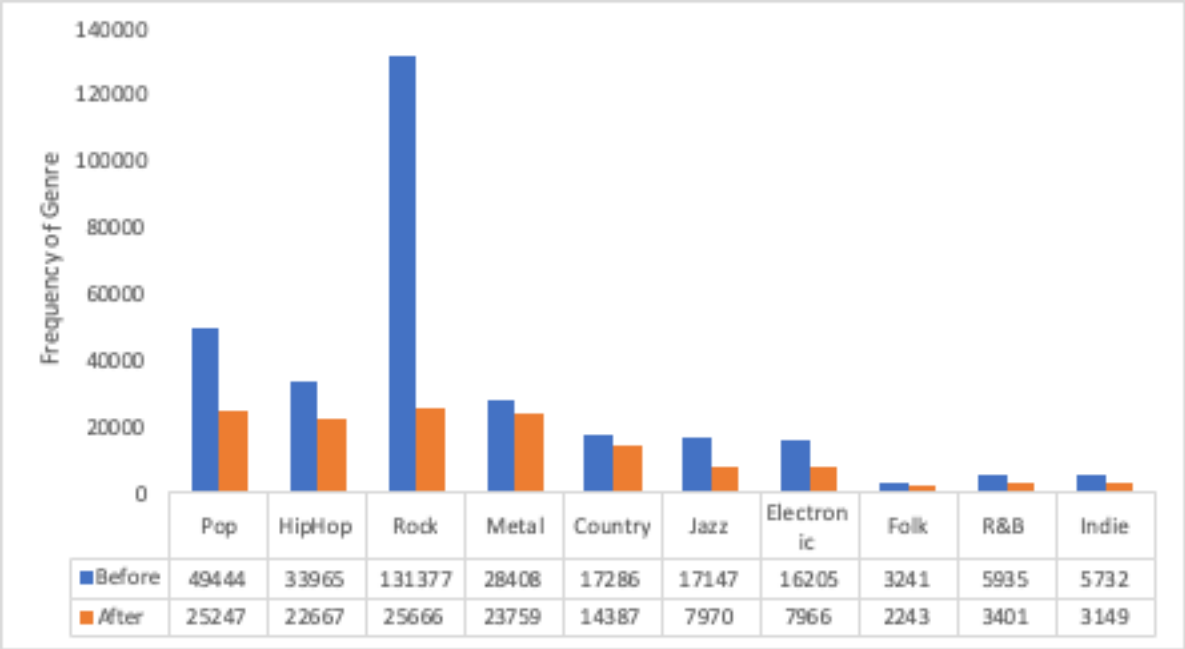
A lot of work has been done on the mood and genre classification of songs. One of the research was done by X.HU and Downiel which was focused on the text of the lyrics [2]. The music retrieval system which processes lyrics information uses lyrics recognition techniques like large vocabulary continuous speech recognition (LVCSR), acoustic model and a trigram language model. Analysis of the lyrics of songs covers different aspects such as structure detection, categorization of themes such as ‘sad’ and ‘happy’. Mood classification plays an important part and the lyrics of the song are very helpful in determining the mood of the song. Different mood types were determined from user tags using lyric features. They also concluded that the combination of lyrics and audio can help in better mood classification. In this project, we will work on the genre classification of songs by analyzing its lyrics using information retrieval techniques.

3. Data-set

The dataset for the project is called “380,000+ Lyrics from MetroLyrics” extracted from kaggle.com [1]. The dataset is labeled and in the form of a CSV file with 6 columns- index, song, year, artist, genre, lyrics. The corpus will have more than 360,000 entries for song lyrics. The input column will be the “lyrics” and “artist” which contains the lyrics of the song and artist name in text. The class column will be the “genre” column. We will divide this dataset for training and validation.

3.1 Dataset Balance

The original dataset contains 362,237 entries, and there are 12 class labels for the Genre- 'Pop', 'Hip-Hop', 'Rock', 'Metal', 'Country’, 'Jazz', 'Electronic', 'Folk', 'R&B', 'Indie', ‘Not Available’, ‘Other’. We removed the data rows with the genre ‘Not available’ and ‘Other’ and calculated the mean word frequency for each class label. We removed the rows from the classes, which had the number of rows higher than mean word frequency. We additionally removed the rows where the row value was null for the lyrics column. After balancing the dataset, the size of the corpus is 136,455. Figure 1 shows the value for each class label before and after the balancing.
The genres for Folk, R&B, and Indie had smaller entries as compared to the other genres, but they were significant enough to keep in the original dataset. To analyze the effect of these entries, we created a down sampled dataset of size 22370 with 2237 rows for each class label to perform the comparative analysis of how significantly it changes the prediction accuracy scores.



	Pop	HipHop	Rock	Metal	Country	Jazz	Electronic	Folk	R&B	Indie
Before	49444	33965	131377	28408	17286	17147	16205	3241	5935	5732
After	25247	22667	25666	23759	14387	7970	7966	2243	3401	3149

Figure 1: Class label distribution before and after balancing

3.2 Pre-Processing the text

On the balanced and down sampled dataset, we implemented the following text processing techniques:

1. Removal of special characters.
2. Word Stemming: We used Snowball-Stemmer for this project. The three major stemming algorithms are Porter, Snowball, and Lancaster. And among those, Porter is the least aggressive algorithm, but Snowball (also known as Porter 2) has slightly faster computation time.
3. Stop word Removal: We used NLTK (Natural Language Toolkit) stop words list to remove the stop words from the text.
4. Created word tokens.

3.3 Feature extraction

To implement feature extraction, we used sklearn's TF-IDF vectorizer which converts a collection of raw documents to a matrix of TF-IDF features. We have used the vocabulary size of 10,000 and used it to define the maximum features for the vectorization, thus we have a maximum of 10,000 word vectors created.

As quoted in sklearn guide [3]- Using the TfidfTransformer's default settings Sklearn library for TF-IDF for calculation of the term frequency, the number of times a term occurs in a given document, is multiplied with idf component, which is computed as:

$$idf(t) = log \frac{1+n}{1+df(t)} + 1,$$

Where *n* is the total number of documents in the document set, and *df(t)* is the number of documents in the document set that contain term *t.* This was originally a term weighting scheme developed for information retrieval (as a ranking function for search engines results) that has also found good use in document classification and clustering.

4. Process Flow Diagram

Figure 2 illustrates the project's functionality low diagram.

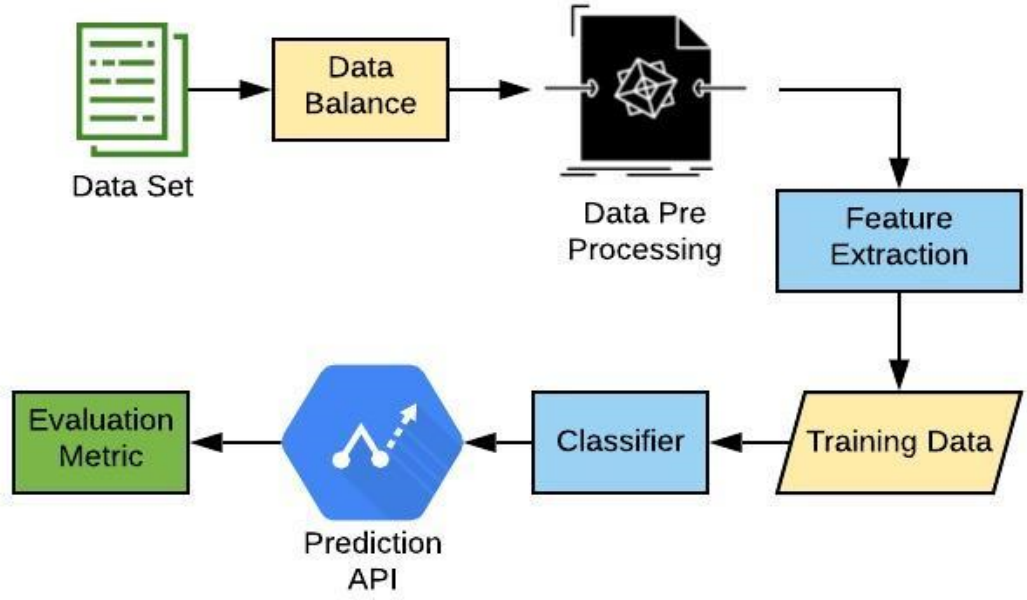


Figure 2: Functionality flow diagram

5. Experiment Design and Results

5.1 Methodology

In this project, we have implemented three classification models on the data-sets (both original and down sampled) that resulted after the text pre-processing and creating the TF-IDF feature vectors. We will perform the classification on both original data-set and down sampled data-set to compare the effect of lower number of data entries in the dataset.

Resulting data-set statistics:

- Size of original corpus- 136,455
- Size of down-sampled corpus- 22,370
- Total Class Labels- 10
- Baseline-
    - Original- 18%
    - Down-Sampled- 10%

5.2 Algorithms Implemented:

In this project, we have implemented both machine learning and deep learning models on two data-sets. The algorithms implemented in this experiment are:

- Naive Bayes
- Random forest- Experiment performed on 30 n-estimators (trees) in the forest.
- Convolutional Neural networks- Implemented 2 dense layers where the first layer is based on the "Relu" function and second layer is based on the "Softmax" function.

5.3 Results

- The performance results for **Naive Bayes**: We have observed that equal distribution of data, as we have implemented in the down-sampled dataset, does impact the accuracy of the model. The confusion matrix illustrates how the prediction is distributed uniformly across the genres.

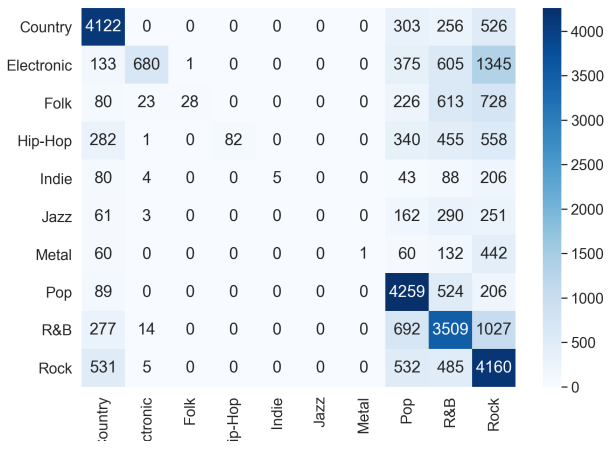
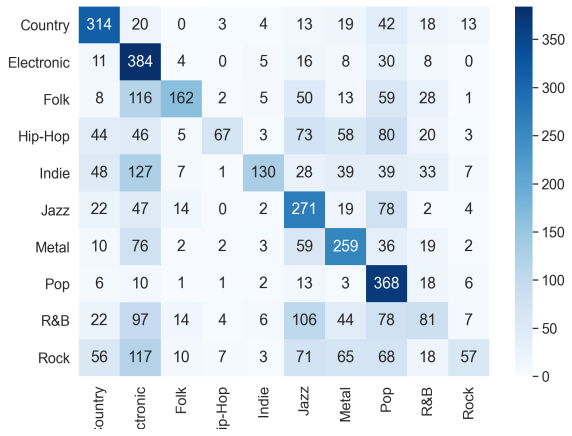
1. **10-Fold Cross Validation:**

Original Data (Baseline- 18%)	Down-Sampled Data (Baseline- 10%)
54.9%	45.3%

2. **F-Score**

Original Data (Baseline- 18%)	Down-Sampled Data (Baseline- 10%)
32.2%	43.09%

3. Confusion Matrix

Original Data	Down-Sampled Data
	

- The performance results for **Random Forest**: Similar to Naive Bayes, random forest also gives better performance which is uniformly distributed across genres as illustrated in the confusion matrix.

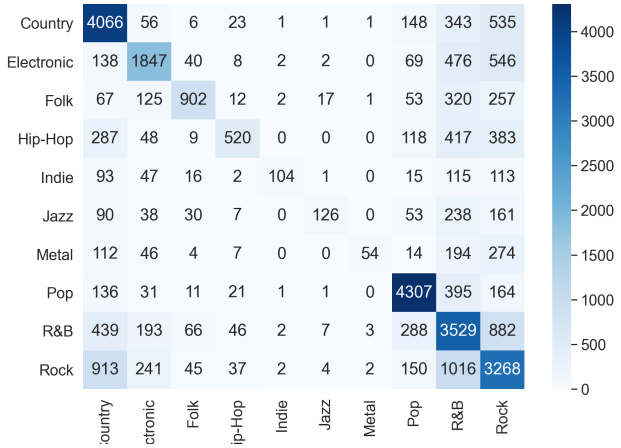
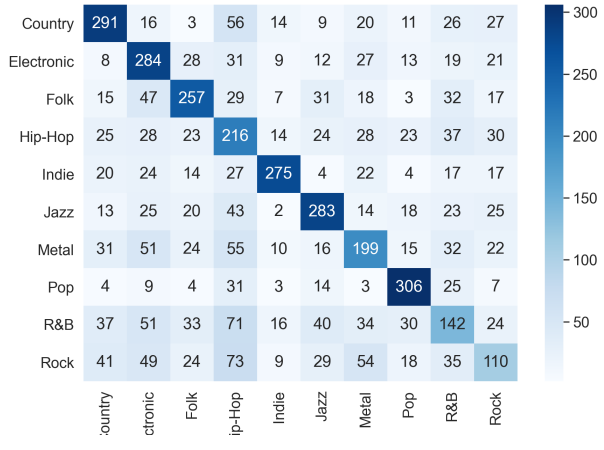
1. 10-Fold Cross Validation:

Original Data (Baseline- 18%)	Down-Sampled Data (Baseline- 10%)
62.1%	50.7%

2. F-Score

Original Data (Baseline- 18%)	Down-Sampled Data (Baseline- 10%)
50.8%	52.06%

3. Confusion Matrix

Original Data	Down-Sampled Data
	

- The performance results for **Convolutional neural network:** CNN performed better for the down-sampled dataset, but the difference was not as dramatic as seen in the Naive Bayes and Random forest. As illustrated in the confusion matrix, the genre “country” had the highest frequency of predictions by the CNN model. We believe that it happened as country music has the most similarity in terms of music and text with other genres.

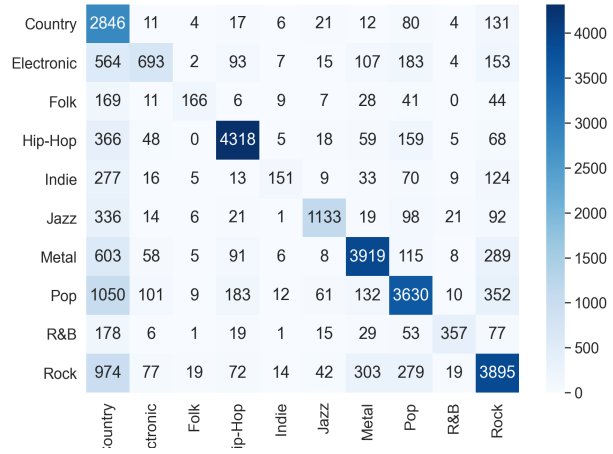
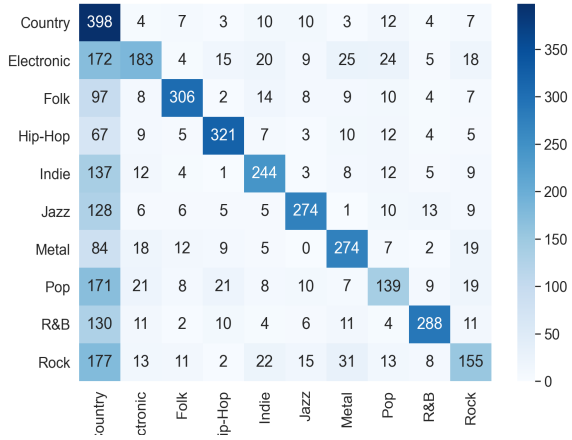
1. Accuracy

Original Data (Baseline- 18%)	Down-Sampled Data (Baseline- 10%)
74.2%	63.4%

2. F-Score

Original Data (Baseline- 18%)	Down-Sampled Data (Baseline- 10%)
65.8%	63.06%

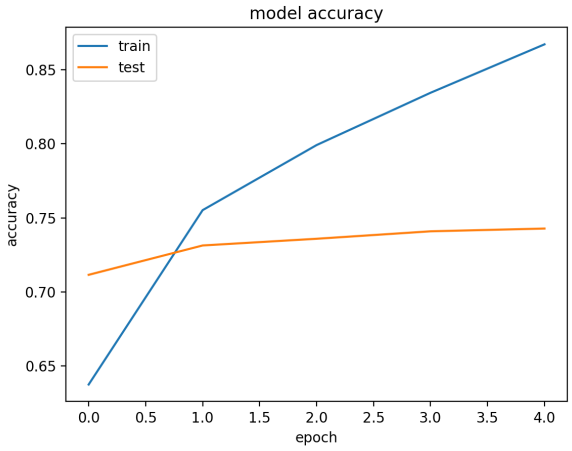
3. Confusion Matrix

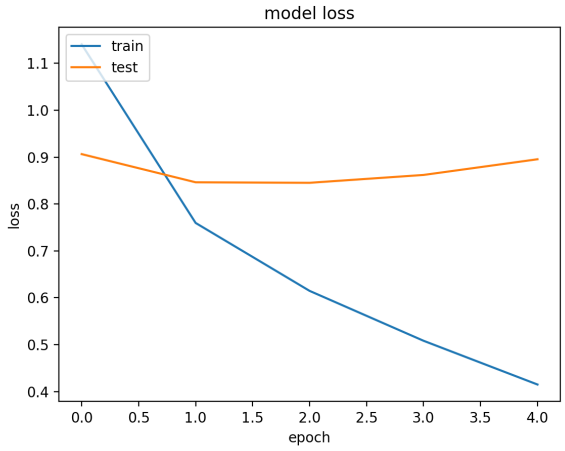
Original Data	Down-Sampled Data
	

4. Accuracy and Loss curves for training and validation:

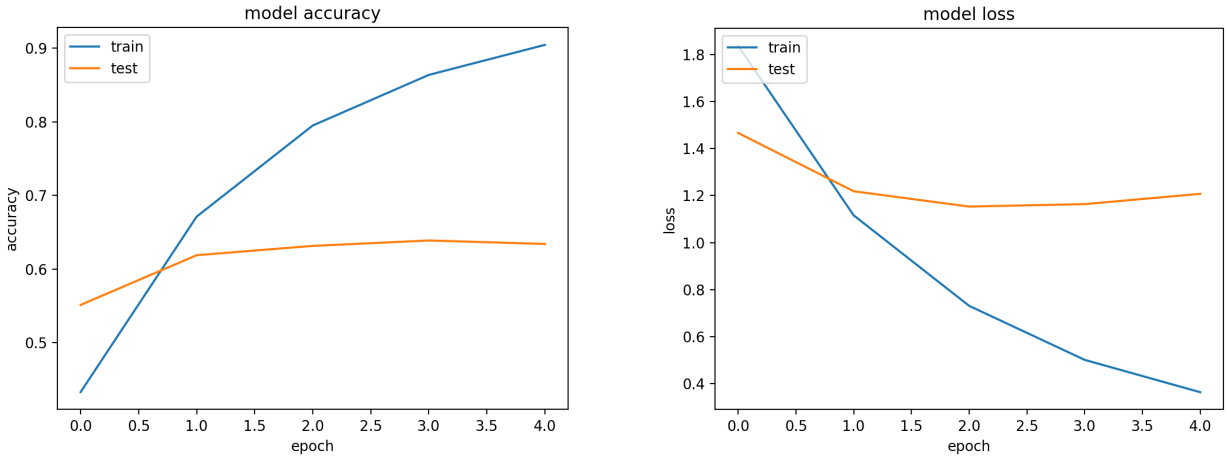
We used 5 epochs for our experiment as we observed that after 5 the loss started to go up and the model started to overfit as illustrated in the charts below:

Original Data: EPOCHS-5





Down-Sampled Data: EPOCHS-5



- **Collated results:**

Baseline- Original Dataset (Org) - 18% || Down-sampled Dataset (DS) - 10%

Naive Bayes				Random Forest				CNN			
10-Fold Cross-Val		F-Score		10-Fold Cross-Val		F-Score		Accuracy		F-Score	
Org	DS	Org	DS	Org	DS	Org	DS	Org	DS	Org	DS
54.9%	45.3%	32.2%	43%	62.1%	50.7%	50.8%	52%	74.2%	63.4%	65.8%	63%

6. Conclusion

In our experiment, we performed the analysis on predicting the genre of the song using the lyrics and artist names in various ML models and across different data distributions. CNN (Deep Learning) resulted in the highest performance for the genre classification, followed by Random forest and then Naive Bayes. The down-sampling of the data resulted in a significant improvement and much uniform results - thus, we can confidently say that data distribution does impact in the given scenario. This experiment helped us understand the ways to pre-process a text document for various ML models and how deep learning performs in comparison to machine learning with the same input.

7. Future Work

The project can take various directions in the future. One being the inclusion of other features given in the text such as- year, song title. The other would be to implement the word embedding such as glove and word2vec and lexicon in order to interpolate the semantic context of the text. Additionally, the

project has a vast scope in terms of analysis on various deep learning techniques and given more time we would have experimented more with the design of the CNN, and its various convolutional layers.

8. References

[1] GyanendraMishra. (2017, January 11). 380,000 lyrics from MetroLyrics. Retrieved from https://www.kaggle.com/gyani95/380000-lyrics-from-metrolyrics/data

[2] X. Hu, J.S. Downie, K. West, A.F. Ehmann, "Mining Music Reviews: Promising Preliminary Results", *Proceedings of 6th International Conference on Music Information Retrieval ISMIR 2005*, pp. 536-539, September 11–15, 2005.

[3] 6.2. Feature extraction¶. (n.d.). Retrieved from https://scikit-learn.org/stable/modules/feature_extraction.html#text-feature-extraction