Music genre identification using lyrics

Georgios Goniotakis (G.Goniotakis@sms.ed.ac.uk)

Abstract

Music genre classification, especially using lyrics alone, remains a challenging topic in Music Information Retrieval. There are various difficulties involved in obtaining and using data for this task. The results presented in related literature have significant issues. In this study, we look at these issues, and present our own experiment set up. We carefully preprocess the data, and classify a large 10-genre dataset of intact song lyrics, experimenting with various parameters and feature engineering. We present a number of findings which might aid in further investigation of this topic. Experimental results show that our set up outperforms the best classification results found in the literature with similar data. We also contribute a new balanced dataset for lyrics-based genre classification.

1. Introduction

Music classification is an important and widely studied problem in Music Information Research (Sturm, 2012; Bogdanov et al., 2016). The emergence of large music collections has created an interesting challenge pertaining to the retrieval, browsing and recommendation of their contents (Schedl et al., 2018). These functions are made possible due to the tagging of entries in these collections, which can be added manually or, more cost-effectively, automatically (Bertin-Mahieux et al., 2011a). Automatic music classification, which produces these tags, is also a potential solution to the cold-start problem, a well-known issue in music recommender systems wherein new songs cannot be recommended until they are tagged (Eck et al., 2008).

Classifying music by its genre is an efficient way to aid the retrieval, browsing and recommendation of music. Music genre labels are useful categories to organise and classify songs, albums, and artists into broader groups that share similar musical characteristics (Lee & Downie, 2004) and are one of the most used descriptors of music (Bertin-Mahieux et al., 2011a). However, genre classification is a difficult problem which is limited to humans' ability to identify a genre (Lippens et al., 2004) and the subjective nature of genre itself (Fu et al., 2011). It is also, however, an important challenge, recently noted to be highly desirable for music recommender systems (Schedl et al., 2018), which are used prolifically in large music streaming services.

Most published genre classification approaches rely on audio sources (Bogdanov et al., 2016; Sturm, 2012). Other sources used for this task include rhythmic features (Mayer et al., 2008), images (Libeks & Turnbull, 2011) and, more recently, visual representations of audio signal in the form of spectrograms (Choi et al., 2017) as well as various combinations of these features (Oramas et al., 2017; Mayer & Rauber, 2011).

The problem we are looking at in this study is the classification of music genre using song lyrics. Given the lyrics of a song, a classifier must identify which genre the song belongs to. This approach to classify music is particularly useful when audio is not a viable option to extract features. Audio is not always available in the right format and is more time consuming to model than text (Mayer et al., 2008). Even though classifiers that use audio surpass those that use lyrics, it has also been shown that the inclusion of lyrics can boost the performance of an audio-based classifier (Mayer & Rauber, 2011). Furthermore, lyrics can be used to identify specific genres which are not necessarily defined by a song's sound but by its lyrics, e.g. 'love songs' or 'Christmas carols' (Neumayer & Rauber, 2007).

In this study, we set out to build a lyrics-based genre classifier and improve upon the state of the art. We identified that the first thing we needed was data, namely the lyrical content of a number of songs and their genres. Unfortunately, research with song lyrics is deeply affected by copyright issues (Tsaptsinos, 2017). The only manageable dataset that we found was the MetroLyrics 380K dataset (Mishra, 2017). The issues this dataset has will be discussed in Section 3.

Automatic music genre classification using lyrics is the subject of multiple studies (Fell & Sporleder, 2014; Tsaptsinos, 2017; Choi et al., 2014; Canicatti, 2016) which show that lyrics are distinctive enough to distinguish genre. Most research studies have either obtained private datasets through agreements, as in Tsaptsinos (2017), or have created their own, such as in Fell & Sporleder (2014) and Canicatti (2016). Canicatti (2016) uses a dataset obtained from MetroLyrics, which is the source of the dataset we have access to (Mishra, 2017).

Through careful review of the literature, we identified a number of issues in the published work which will be discussed in the next section. Keeping these issues in mind, we set out to replicate experiments described in the literature and to build on top of current work to achieve a better classification performance for this task.

In this paper, a critical analysis of the state of the art is

presented from the perspective of experimental set up. We run experiments described in this related work and document the process including the difficulties and doubts we came across. Through this experimentation, we are able to present a number of contributions in this paper. We present classification results which surpass those conducted with similar data and which use more training samples than any other study the authors are aware of. An advanced preprocessing module for lyrics is also proposed. Additionally, we contribute a new balanced dataset for lyrics-based music genre classification and demonstrate its eligibility through the presentation of experimental results.

Section 2 discusses work related to this study. In Section 3 we describe the dataset used and present findings from exploratory data analysis. We also discuss the preprocessing we applied to the dataset. In Section 4 we present a description of the models and metrics used in the experiments, which are described in Section 5, along with results. Finally, we conclude with a discussion on our findings and future work in Section 6.

The current study serves as an easy-to-follow guide aimed towards an audience with a broad level of expertise in the music retrieval domain. It includes thorough details about our research, implementation and experimentation methodologies. Graphs and tables showcase the characteristics of the dataset and results of our experiments. Sufficient information is provided to allow reproducibility by people interested in this field.

2. Related work

McKay & Fujinaga (2006) emphasised the importance of automatic genre classification in their paper. The findings from the survey by Lee & Downie (2004) show that users are more likely to search songs by genre than by any other filter such as artist or recommendation. They indicate that lyrics play a strong role in genre classification.

Luštrek (2006) surveyed the techniques for automatic genre identification from text in literature, including traditional methods. Traditional methods include feature engineering, varying from analysing parts of speech to considering word lengths. They argue that using traditional methods, which at the time the study was conducted were the most researched, yield best results in genre classification, however, more work is required for difficult tasks and datasets.

One of the pivotal works in lyrics-based genre classification is that of Mayer et al. (2008). In their paper, they identified lyrics-based genre classification as a difficult task due to differences between lyrics and traditional text - mainly that lyrics are structured in verses and that the frequencies of the words matter more. A semantic and structural analysis of song lyrics can be found in Mayer & Rauber (2010).

Mayer et al. (2008) also presented a set of features which model lyrics and combined them with bag-of-words approach. They used text statistics, part-of-speech analysis and rhymes to model text and compared k-NN, Naive Bayes, SVM and Decision Trees, achieving best performance with SVM with an accuracy of 33.47%. They note that although the accuracies presented are lower than comparable results based on audio, this classification may be complementary and improve the performance of audio-based classifiers. In fact, in their later paper (Mayer & Rauber, 2011), they combine their lyrics features with audio features to present a better classifier. They achieve a 74.08% accuracy on a "large" dataset, a 6% improvement from using only audio. Therefore, improving lyrics-based classification accuracy is important to boost the overall accuracy of the combined approach.

More recent work has endeavoured to improve the performance of lyrics-based genre classification. Fell & Sporleder (2014) use n-grams along with other hand-crafted features to classify among 8 genres. Ying et al. (2012) use POS tags to classify among 10 genres with a highest accuracy of 39.94%. Tsaptsinos (2017) conducted experiments with deep learning models to achieve a top accuracy of 49.77% using an LSTM across 20 genres.

These results are difficult to compare due to a number of factors. Firstly, the studies use different datasets, which also differ in size. Secondly, different studies focus on a distinct number of genres. Classification results in this case cannot be directly compared as the classes, as well as the number of classes, vary (Demšar, 2006). In addition to this, most results are reported in terms of accuracy. This evaluation metric has be shown to be an insufficient way to measure classification performance (Sturm, 2013). This is especially when the dataset is unbalanced (Provost & Fawcett, 2001), as is the case in Tsaptsinos (2017) and Fell & Sporleder (2014).

In the study by Canicatti (2016), a dataset using MetroLyrics is constructed. The dataset used for our experiments (Section 3) is also sourced from MetroLyrics. Canicatti (2016) explain that the lyrics used are inputted by users, meaning their data might have wrong inputs and typos, which is significant if one uses approaches like tf-idf (Section 4). They experiment with different models such as Decision trees, Random Forest, k-NN and Naive Bayes over 2500 songs to achieve a highest accuracy of 51.69% with Random Forest among 5 genres. Although our lyrics come from the same source, the experiments in this study are conducted on a much larger dataset and perform classification over twice the number of genres.

As mentioned in the introduction, lyrics-based research is plagued with copyright issues, making it difficult to obtain data. Mayer & Rauber (2011)'s "large" dataset only consists of 3000 songs, (Ying et al., 2012), 600. Whilst Howard et al. (2011); Canicatti (2016); Fell & Sporleder (2014) all created their own datasets for their research, none of them are available for public use. The lack of publicly available datasets and quality of the data (Section 3) motivated us to create our own dataset, which we present as part of our work and whose benefits are described in Section 6.

3. Data

The dataset we use contains 380,000 songs (Mishra, 2017). It includes interesting features such as title, year, artist and, most importantly for our application, genre. The dataset contains web-scraped song lyrics from the MetroLyrics website (MetroLyrics), as in (Canicatti, 2016) and (Ying et al., 2012), and is available on Kaggle under a Creative Commons (CC BY-SA 4.0) license. Despite this license, this dataset contains web-scraped lyrics, violating the Intellectual Property regulations, and should probably not be located on the web. For the purpose of having a common comparison scale with other studies, we will use this dataset for our research, but will not upload any modified version of it on the web.

Exploratory Data Analysis (EDA)
The dataset consists of 12 different genres: Pop, Hip-Hop, Not Available, Other, Rock, Metal, Country, Jazz, Electronic, Folk, R&B and Indie. In terms of class distribution, the rock genre massively outnumbers the others as it comprises of **36%** of all the records. The folk category is the most under-represented one with just 2243 samples. Two categories, "Not Available" and "Others", consist of mixed genres and are therefore not included in our Machine Learning pipeline. The records of those two categories represent **14%** of all samples in our dataset. **95,680** records do not have lyrics and so are also removed from our data collection.

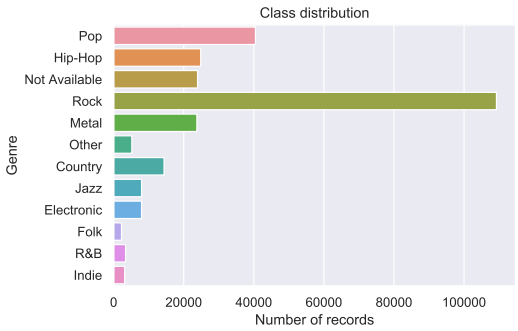

Figure 1. Class distribution

This significant imbalance in class distribution creates a limitation for the application of Machine Learning to this data. Our options when it comes to class imbalances are either oversampling (through a data augmentation technique like Synthetic Minority Over-sampling Technique (SMOTE) (Chawla et al., 2002)) or undersampling using some industry-standard library such as imbalanced-learn (Lemaître et al., 2017). Oversampling artificially creates more samples by duplicating some of the records in under-represented classes. This is undesirable for our data since it would lead to our classifier believing certain words to have a strong indication of a certain genre. Hence, we tried to undersample the data which left us with just **22,430** samples, since undersampling matches the number of records of the minority class. Fitting a basic Logistic Regressor produced low scores for both the accuracy and f1-score of **0.333** and **0.321** respectively. Trying a baseline model did not produce acceptable results, so like other studies (Tsaptsinos, 2017; Fell & Sporleder, 2014), we decided to move on with using the dataset as is (without rebalancing the classes).

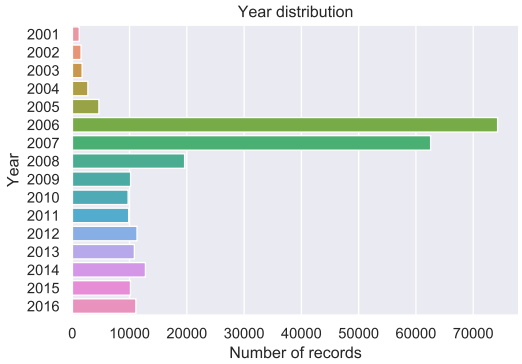

Figure 2. Year distribution

Regarding the year distribution, the classes are not balanced either. **50%** of the songs contained in this dataset were produced in the years 2006 and 2007. The oldest song of the collection was produced in 1968 and the most recent in 2016.

A visualisation (Figure 3) of the most commonly occurring words in the lyrics provided us with an indication that there was an issue with the quality of the data. We can easily see that the words "Beyon" and "Beyonce", which refer to the eponymous artist, appear with a huge frequency in our dataset. Upon deeper investigation of the data, we discovered the existence of tags with the names of the artists enclosed (i.e. [Beyonce]), used to indicate the person singing the ensuing lyrics. Related studies do not refer to such a characteristic, which could mean such elements were passed into Machine Learning pipelines of other studies. Such information could potentially create a bias to the Machine Learning model by making it believe that the word "Beyonce" is highly correlated with "Pop", which is the genre that Beyonce has mainly produced songs in. Additionally, we can see that a very common word is "Verse", which denotes a structural element of the song, irrelevant to lyrical content. This could contribute to noise in the overall data. Here we demonstrate how a simple visualisation such as Figure 3 can expose problems with the data and why "knowing your data" is a critical step in the overall process.

Data Preprocessing
In their study, Fell & Sporleder (2014) state that the field of lyrics-based classification "would benefit from the development of more sophisticated methods for cleaning, processing and analysing song texts".

From our initial data analysis it became obvious that the lyrics contained a lot of noise. A part of that noise also included information like the name of the artist and song, which were giving an unfair advantage when Machine Learning was applied to the dataset. As a first step (as in other studies) we removed the two categories ("not available" and "other") consisting of songs of either mixed or

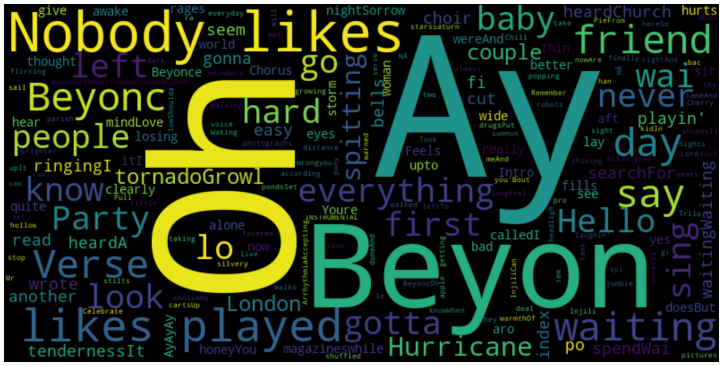

Figure 3. Word cloud of top 200 words

no genre. Moreover, duplicate songs (songs with the same artist and title) were removed as they can artificially increase the importance of some features during the encoding and Machine Learning steps. Artists with songs from multiple genres were also identified and removed from our collection.

Firstly, we created a routine which removes all the stop words from our lyrics. The list of stop words that we used is the NLTK's list with common English stop words (Bird et al., 2009). By performing stop word removal, we removed unnecessary noise from our data which when vectorizing our data will result in fewer dimensions and faster computations. The total number of words removed from the dataset is **24 million** words. Alongside that, we removed unnecessary symbols like trademark signs, digits (numbers) and converted accented characters to their non-accented version (i.e. é to e). In total we removed **110,000** symbols, and converted almost **10,000** accented characters.

As discussed earlier, we observed tags containing unfair information inside our lyrics. The literature fails to identify such blocks of information which potentially means that only the structure tags included in the lyrics (such as [Verse] anf [Chorus]) were identified and removed in other studies. We were able to identify and remove **116,000** tags from our data collection, removing noise and avoiding unnecessary bias.

During our initial exploration, we found a considerable amount of typos (spelling errors) which were due to the nature of user-inputted lyrics on the MetroLyrics (MetroLyrics) web page. In order, to deal with these we utilized a simple yet efficient language probability (Norvig, 2007) model produced by Peter Norvig (Director of AI research at Google). However, after further testing, we saw that this model can be good at fixing spell errors but it is also trying to amend correct words (i.e. convert "I" to "a"). Of course, that was expected given the simplicity of its implementation.

To further normalize our lyrics we used three large collections of contractions (ain't - are not), abbreviations (approx. - approximately) and slang (ima - I am going to) words to expand and replace problematic words in our dataset. Using these utilities we were able to expand **3.5 million** contractions and replace **680,000** slang words.

Non-English songs can also introduce a significant bias. As an example, **100%** of the Spanish songs on our dataset belong to the 'Pop' genre. In order to track and remove songs of non-English languages, we utilized the language detector module of the state-of-the-art Natural Language Processing (NLP) library spaCy. Using it we were able to identify and remove **14,000** non-English songs. This leaves us with a dataset containing **187,186** unique records produced by **14,543** artists.

As a last step, we used the Snowball stemmer shipped with NLTK to stem all the words of our lyrics.

Extensive testing with trial lyrics was performed to establish the correct order of applying the various preprocessing methods to the dataset. Hence, we are certain that preprocessing steps do not conflict with each other.

Dimensionality Reduction

A bag-of-words model can produce sparse matrices (matrices whose contents are largely zeros), which can hinder the performance of our Machine Learning algorithms and make the overall computation expensive. Hence, we experimented with different approaches of dimensionality reduction. Besides trying more standard techniques like filtering by Pearson correlation, Chi-2, and Recursive Feature Elimination (RFE) (Guyon et al., 2002), we also experimented with different industry-standard ways of filtering features of low importance.

Firstly, we applied TF-IDF (Section 4). By varying the maximum number of features (using a Grid Search technique from 1000 to 80,000 words) and fitting a baseline Logistic Regressor, we obtained the best performing configuration (**13,000 words**).

Another approach we tried used SelectFromModel, a meta-transformer model in scikit-learn. By vectorizing the text, passing the features inside a Linear Support Vector Classifier which uses L1 penalization and passing the output to SelectFromModel, we can retrieve the non-zero coefficients and hence, our resulting sample set.

Out of all dimensionality reduction approaches tested, the

set produced by the SelectFromModel function coupled with a Linear Support Vector Classifier produced the best accuracy and is going to be used in all of our shallow models. The final number of features (words) used during the training of our algorithms was approximately **8000 words**.

Another approach we experimented with was Word Embeddings (Mikolov et al., 2013). We used the Global Vectors for Word Representation (GloVe) (Pennington et al., 2014) produced by researchers at Stanford University. Word embeddings, when coupled with neural network architectures like LSTM have the potential to build strong classifiers. We used word embeddings to train the LSTM and GRU models presented in our study, as described in Section 6. The dimensionality of data passed through to these models depends on the lexical coverage of these word embeddings over the lyrics.

Feature Engineering

In terms of feature engineering, we created a basic list of combinatorial NLP from the lyrics: number of words, unique number of words, number of characters, number of punctuation characters, number of uppercase words, number of title case words and average word length. Additionally, we took advantage of our thorough data preprocessing routine to build interesting features such as: number of tags, number of contractions, number of slang words, number of digits, number of accented characters, number of new lines and number of stop words for each record.

To detect any significant correlations the genre class was transformed using One Hot Encoding (OHE) (Section 4) and the correlation values between features were calculated and visualized on a heat map. Observing the correlation matrix, we cannot see any significant dependency besides a very interesting one between the number of stop words and the hip hop category (**0.46** correlation). The correlation matrix can be found in the Appendix I.

Our framework is able to perform full preprocessing and creation of combinatorial features (feature engineering) in just **0.1 second** per record.

4. Methodology

Text Representation

One Hot Encoding: In order for textual data to be interpreted by our models, we need to represent it numerically. For the classes, we are converting each genre into a different representation. We used the One Hot Encoding (OHE) approach to create a binary matrix where 1 means that the song belongs to a specific genre and 0 that it does not. The reason that we preferred this instead of another encoding methodology like label encoding in which the categories are represented by positive integer numbers (i.e. 1-6) is that different genres do not actually represent a change in quantity. Thus, encoding the categorical features using a label encoding approach would trick our machine learning models. Representing our classes using one hot encoding also allows us to examine, using correlation values, how each of our combinatorial features affects the target (genre) (Figure 6).

TF-IDF: In order to prepare the data for use in the machine learning models Logistic Regression, Random Forest, Boosting models and MLP, we use TF-IDF to vectorise the lyrics. Tf-Idf (Term Frequency times inverse document frequency) represents a lyrics corpus by using the unique words in the corpus' dictionary as descriptive features (Nyberg, 2018). The weights of the feature depend on how common the words is. This method of modelling lyrics is both efficient and simple (Ramos et al., 2003), especially in this context with music lyrics due to the nature of the culture of music McKay & Fujinaga (2006).

Tf-idf of a term t is calculated as:

$$tfidf(t) = tf(t) \times idf(t) \tag{1}$$

where:

$$TF(t) = \frac{\text{No. of times t appears in the document}}{\text{No. of terms in the document}} \tag{2}$$

and

$$IDF(t) = log_e \frac{\text{No. of documents}}{\text{No. of documents with term t}} \tag{3}$$

The weakness of this approach stems from the fact that the database entries are created by internet users and are not 100% accurate. If not enough preprocessing is done to fix user typos, these words' inverse frequency score would be higher although they are common words. For example, Beyon, Beyonc and Beyonce have seperate tf-idf scores, and therefore having the word Beyonce appears to be a more unique feature than it is for the lyrics.

Models

The shallow models were implemented using scikit-learn (Pedregosa et al., 2011) and the deep learning models using Keras (Chollet, 2015).

Logistic Regression: Multinomial logistic regression was used as our baseline due to the model's simplicity and efficiency. We also use this model to evaluate whether our advanced preprocessing affects classification performance (Section 6).

Boosting: Boosting enhances the performance of a decision tree classifier. It attempts to reduce misclassifications, combining weak classifiers to create a strong classifier which performs well on data previously misclassified. It achieves it by putting more weight on difficult data (Freund et al., 1999). It was included in our experiments as it has been shown to perform well on text classification tasks (Bloehdorn & Hotho, 2004; Jafarpour et al., 2018). Two types of boosting were used: AdaBoost and Gradient Boosting. The difference between the two algorithms is in how they create weak classifiers. AdaBoost assigns more weight to misclassified instances, and less weight to correctly classified instances (Hastie et al., 2009). Gradient Boosting minimises the error of the strong classifier using a gradient optimisation process (LeCun et al., 1998).

Random Forest: This algorithm is intrinsically suited for multi-class problems. It builds an ensemble of decision trees, the number of which is supplied as a parameter and averages the results. It performs well with many features, as it searches for the best feature. We used Random Forest with 500 trees, as this was the best performing model and setting in Canicatti (2016), which used data from the same source as the data in this study.

Long-Short Term Memory: A Long-Short Term Memory (LSTM) Network is a Recurrent Neural Network with LSTM units (Gers et al., 1999). This type of network is used on textual data because of its ability to store information over arbitrary time periods. Since lyrics are made up of a sequence of words, the historical context of inputs is important, as each word depends on the ones that came before it for meaning to be conveyed .

Our LSTM Network is made up of an embedding layer, 20 LSTM units and a Dense layer (a fully connected network with softmax activation function). It uses categorical cross-entropy as a loss function and adam as the learning optimizer. Each set of lyrics is vectorized to 300 elements. For our LSTM model, we use pretrained word embeddings from Global Vectors for Word Representation (GloVe)(Pennington et al., 2014). These are trained using statistics about word co-occurrences, based on the hypothesis that words which occur together are more likely to have associations with each other.

Gated Recurrent Unit: Gated Recurrent Units (GRUs) are a structure in Recurrent Neural Networks (RNNs) designed to capture long-term dependencies in spite of the gradient-vanishing problem of RNNs (Chung et al., 2014). This is done by adding intermediate states in between hidden states of an RNN. A GRU determines how important the previous state is to the next hidden state and also to the memory. It has been shown to perform well with sequence models, and has even been shown to outperform LSTM in text categorisation (Wang, 2018).

Evaluation Metrics

In the following sections we will describe metrics in the terms of True Positives (TP), True Negatives (TN), False Positives (FP) and False Negatives (FN).

In the case of our classification problem, the definitions are as follows for the example of the 'Pop' class:

- True positive: Pop song correctly identified as pop
- False positive: Non-pop song incorrectly identified as pop
- True negative: Non-pop song correctly identified as not pop (classified as another genre)
- False negative: Pop song incorrectly identified as not pop (classified as another genre)

Each metric presented takes an overall average of the collective performance of all classes according to that metric.

Accuracy: The Accuracy of a model in this problem is the percentage of songs in the data that were classified correctly. However, as discussed in Section 3, we are dealing with an imbalanced dataset, which means that a measure of accuracy is not really indicative of the effectiveness of the classifier. This refers to the accuracy paradox (Valverde-Albacete & Peláez-Moreno, 2014). We therefore use other, more robust metrics, which will be described below.

Precision and Recall: Precision of a genre is the proportion of songs classified as being in a genre that are actually in that genre. A high precision value reflects a low false positive rate. In our example this would mean that very few non-pop were recognised as being pop. The *Recall* is the proportion of pop songs that are correctly classified (Powers, 2011). A high recall rate in our case would mean that very few pop songs were classified wrongly. We can simply quantify these using equations 4 and 5.

$$P = \frac{TP}{TP + FP} \tag{4}$$

$$R = \frac{TP}{TP + FN} \tag{5}$$

F1-Score: By definition the F1 Score is another measure of a model's accuracy. It is a combination of both the precision and the recall (6). An F1 score, like precision and recall, reaches its best at 1 and worst at 0.

$$F_1 = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{6}$$

Confusion Matrix: A confusion matrix is a table showing the number of correct and incorrect predictions for each class. In the case of a misclassification, it shows what the incorrect prediction was. It provides insight into the types of errors being made by a classifier. This is particularly useful since we have an unbalanced dataset, as it allows us to investigate classifications of majority and minority classes. The confusion matrix for our best performing model can be found in Figure 9.

For all the experiments that we ran, we used a 60/20/20 percent split for the training, validation and test set respectively.

5. Experiments

Using the models and techniques presented above, we developed a Machine Learning benchmarking framework. The framework handles most of the common issues during a Machine Learning experimentation procedure like: debugging, checkpointing, hyper-parameter optimization, storing the final model, and producing detailed performance analytics. The only thing that the user has to do is to load their data into the memory and then, by using just one line of code, define the parameters of their experiment (model, word representation method, and path for storing the model and the performance analysis). The user can provide the routine with a list of hyper-parameters (batch size, number of epochs, number of layers, etc) and the routine automatically logs and compares the best performing combination by using the accuracy and F1-score of the validation set. At

the end of the experiment, the script produces a detailed report which includes the Accuracy, F1-score, Precision, Recall and Confusion Matrix for the training, validation and test sets. Our final test suite is consisting of more than 2000 experiments ranging from a couple of minutes to hours in execution time. We compiled a list of the experiments with the highest performance and we present them in Table 1.

Model	Accuracy	F1	Precision	Recall
Logistic Regression (BP[1])	0.553	0.272	0.539	0.249
Logistic Regression	0.612	0.313	0.654	0.284
AdaBoost	0.552	0.222	0.437	0.216
Gradient Boosting	0.575	0.249	0.637	0.228
Random Forest	0.624	0.356	0.720	0.302
MLP	**0.632**	**0.407**	**0.556**	**0.367**
LSTM	0.589	0.287	0.564	0.270
GRU	0.591	0.288	0.452	0.272

Table 1. Experiment results (Test Set performance)

For all the Deep Learning Models used, in order to avoid overfitting, we used Early Stopping with a goal of maximizing validation accuracy, a patience of 10 and a gap of improvement (Δ) per epoch of 0.2%.

6. Discussion

Starting off with a baseline Logistic Regression model with data which underwent basic preprocessing (bad character and stop word removal, and stemming), we experienced similar results to those produced by state-of-the-art experiments with the same data source (Canicatti, 2016), with a generous **4%** improvement. Additionally, we wanted to see if our advanced preprocessing methodology would enhance results. Fitting a Logistic Regressor on data that were thoroughly cleaned using our advanced preprocessing routine, yielded an improvement of **6%** over the basic one.

Using our thorougly sanitized dataset, we continued our experimentation by exploring ensemble methodologies, in particular Random Forest, which currently holds the best accuracy across all different studies. By using the same configuration (500 trees) as the best performing model found in the literature (Canicatti, 2016), we produced an F1-score of **35.6%**.

Knowing that boosting models can handle imbalances better by focusing on the misclassified samples, we tried AdaBoost and Gradient Boosting which produced F1-scores of **22.2%** and **24.9%** respectively which were not an improvement on our existing recorded performances.

We wanted to explore how a fully-connected network like a Multi-Layer Perceptron (MLP) functions on such a dataset. Leveraging the ease of running experiments with hyper-parameter tuning with our benchmarking framework, we run a set of experiments by applying a Grid Search on parameters such as the batch size: [8-1024], number of nodes on hidden layer: [1-1000] and number of epochs [1-200]. The best performing configuration was: **8000 words, 1024 batch size, 132 nodes** on a hidden layer and **9 epochs** which produced an F1-score of **40.6%**.

LSTM and GRU were also used as part of our experimentation but extensive hyper-parameter optimization was not tried on them. Using word embeddings and 100 epochs produced an F1-score of **28.7%** for LSTM and **28.8%** for GRU.

To sum up, our custom fully-connected network managed to improve the current state-of-the-art by approximately **12%**. However, we are aware that since we are using deep neural networks there is a high risk of overfitting the data. Utilizing regularization and/or k-fold cross validation, we would be able to quantify and alleviate the phenomenon. For readers that are seeking for a deeper insight on the performance of our model, we are attaching the performance report for the test set on the second Appendix chapter.

Empowered by the difficulties presented on this study such as data quality and structure of the 380K dataset (class imbalances, a lot of unlabelled data and inadequate year distribution), we are proposing the creation of a better-overall dataset which will enable the researchers interested in the music retrieval domain to perform quality research. Taking into account the limitations of the 380K dataset, we compiled a flexible data collection which can be loaded in the memory of a standard personal computer and allows rapid experimentation without sacrificing important information.

In order to collect all this information, we created an efficient script in Python programming language which is able to obtain the metadata and lyrics of over 300 thousand songs in less than 5 hours. To achieve this, the script detects the maximum available threads of a machine and then performs retrieval of songs' metadata and annotation with lyrics using multi-threading (multi-processing). The only thing that the user has to do in order to operate it is to obtain two free API keys (one for Spotify API and one for Genius API) and set their values inside a configuration file. In this way, the researchers interested in this dataset can acquire it without us publishing copyrighted work (lyrics) on the web.

Our dataset consists of approximately **175 thousand** songs originating from **14.569 unique artists**. Our dataset includes **9 genres**: [pop, rap, rock, metal, country, jazz, indie, folk and electronic] with much more harmonic distribution than the 380K dataset.

Our dataset covers a wide range of years (1999-2018) and provides an even distribution across them.

Our dataset is balanced in terms of classes and years and it can be used for a plethora of different music retrieval tasks such as exploring trends on lyrics given a specific period or genre, build genre, artist or year predictors, etc. As an indication of performance, we fitted a Random Forest of 500

[1]Basic Preprocessing

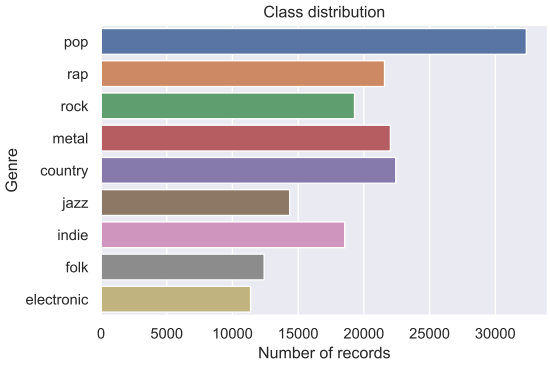

Figure 4. Class distribution (proposed dataset)

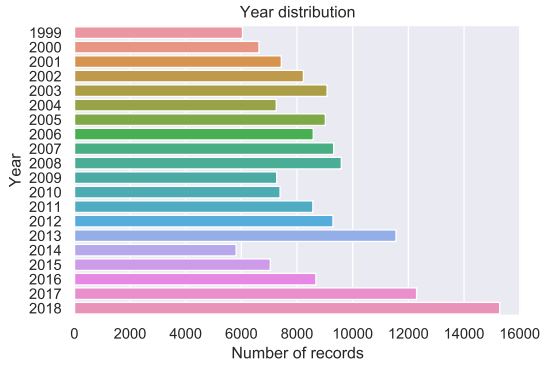

Figure 5. Year distribution (proposed dataset)

trees to a cleaned version of this dataset which produced an accuracy of **41.3%** and an F1-score of **36.2%**.

7. Conclusion

On this study we used a 380K song dataset in order to develop a music genre classifier which utilises lyrics to perform a prediction. After a careful initial evaluation of the dataset and an examination of the literature, we were able to identify the problematic points of other studies and to develop a solution with these points in mind. Our solution uses creative preprocessing and feature engineering methodologies to prepare the data and create powerful combinatorial features. Our Machine Learning benchmarking framework is a powerful tool which allows fast experimentation by supporting a number of shallow and deep learning architectures. Combining all of those we were able to build a solution which demonstrates a distinct improvement over state-of-the-art classifiers. Identifying the limitations of the dataset allowed us to design and implement a new evenly distributed (in terms of classes and years) dataset which will be a useful contribution to the ever-growing field of Music Information Retrieval.

The main directions that an individual or a group could explore can be found below.

Data Acquisition
Collect even more data. MusixMatch API is an enterprise-level API for retrieving songs metadata along with their lyrics (MusixMatch). The lyrics are thoroughly reviewed by human operators to ensure quality. During the course of this project, we contacted the company with the query of providing us a free educational license but unfortunately we did not get any response. The Million song dataset (Bertin-Mahieux et al., 2011b) serves as a good solution for implementing a reasonable model but due to its enormous size (280 GB) we could not load it in some instance to analyze it.

Data Preprocessing
Instead of removing the non-English songs, a Translation API could be leveraged in order to translate the lyrics of a given song to English. The latter is not the best possible solution but it could make the classification pipeline even more complete by adding support for non-English languages. Regarding spelling mistake correction, we saw that just by using a simple language probability model can have less than desirable effects. Alternatively, a deep learning-inspired (Ghosh & Kristensson, 2017) approach could be used. Using different word embeddings such as (Joulin et al., 2016) which uses subword segments would result in positioning mispelled words close to their correct version and as a consequence can produce similar results to spell error correction.

Feature Engineering
Creation of additional combinatorial features such as Part of Speech (POS) tags, Named Entity Recognition (NER) tags, song structure elements (if songs has a chorus or not), number of repeated words in a sequence (i.e. oh oh oh), ratio of uncommon to common words using online word dictionaries would be some of the recommendations regarding feature engineering.

Machine Learning & Hyper-Parameter Tuning
Test more shallow and deep learning architectures. In terms of deep learning architectures we met CNNs (LeCun et al., 1998) and HANs (Yang et al., 2016) in the literature that would be interesting to examine how they perform on the given dataset. It is almost certain that using BERT (Devlin et al., 2018), which is a state-of-the-art Neural Text Classifier recently open-sourced by Google researchers will achieve a stellar performance. BERT has proved that is more than capable of handling imbalanced classes with ease. Once achieved the best performance, an interesting case would be to try frameworks which leverage either Neural Architecture Search (NAS) (Zoph et al., 2018) or the even more recent Efficient Neural Architecture Search (ENAS) (Jin et al., 2018) like H20.ai, Auto-Keras, Google's AutoML to explore even more efficient architectures which maximize performance. We tried to use Auto-Keras on our machines but due to the lack of resources the framework could not function properly.

The improvements proposed above have to be applied to both the 380K and our dataset to quantify their importance using our Machine Learning benchmarking framework.

References

Bertin-Mahieux, Thierry, Eck, Douglas, and Mandel, Michael. Automatic tagging of audio: The state-of-the-art. In *Machine audition: Principles, algorithms and systems*, pp. 334–352. IGI Global, 2011a.

Bertin-Mahieux, Thierry, P. W. Ellis, Daniel, Whitman, Brian, and Lamere, Paul. The million song dataset. pp. 591–596, 01 2011b.

Bird, Steven, Klein, Ewan, and Loper, Edward. *Natural language processing with Python: analyzing text with the natural language toolkit*. " O'Reilly Media, Inc.", 2009.

Bloehdorn, Stephan and Hotho, Andreas. Boosting for text classification with semantic features. In *International workshop on knowledge discovery on the web*, pp. 149–166. Springer, 2004.

Bogdanov, Dmitry, Porter, Alastair, Herrera, Perfecto, and Serra, Xavier. Cross-collection evaluation for music classification tasks. In *17th International Society for Music Information Retrieval Conference (ISMIR 2016)*, 07/08/2016 2016. URL http://hdl.handle.net/10230/33061.

Canicatti, Anthony. Song genre classification via lyric text mining. In *Proceedings of the International Conference on Data Mining (DMIN)*, pp. 44. The Steering Committee of The World Congress in Computer Science, Computer Engineering and Applied Computing (WorldComp), 2016.

Chawla, Nitesh V, Bowyer, Kevin W, Hall, Lawrence O, and Kegelmeyer, W Philip. Smote: synthetic minority oversampling technique. *Journal of artificial intelligence research*, 16:321–357, 2002.

Choi, Kahyun, Lee, Jin Ha, and Downie, J Stephen. What is this song about anyway?: Automatic classification of subject using user interpretations and lyrics. In *Proceedings of the 14th ACM/IEEE-CS Joint Conference on Digital Libraries*, pp. 453–454. IEEE Press, 2014.

Choi, Keunwoo, Fazekas, György, Sandler, Mark, and Cho, Kyunghyun. Convolutional recurrent neural networks for music classification. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 2392–2396. IEEE, 2017.

Chollet, FranÃğois. keras, 2015. URL https://github.com/fchollet/keras.

Chung, Junyoung, Gulcehre, Caglar, Cho, KyungHyun, and Bengio, Yoshua. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014.

Demšar, Janez. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine learning research*, 7(Jan):1–30, 2006.

Devlin, Jacob, Chang, Ming-Wei, Lee, Kenton, and Toutanova, Kristina. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

Eck, Douglas, Lamere, Paul, Bertin-Mahieux, Thierry, and Green, Stephen. Automatic generation of social tags for music recommendation. In *Advances in neural information processing systems*, pp. 385–392, 2008.

Fell, Michael and Sporleder, Caroline. Lyrics-based analysis and classification of music. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pp. 620–631, 2014.

Freund, Yoav, Schapire, Robert, and Abe, Naoki. A short introduction to boosting. *Journal-Japanese Society For Artificial Intelligence*, 14(771-780):1612, 1999.

Fu, Zhouyu, Lu, Guojun, Ting, Kai Ming, and Zhang, Dengsheng. A survey of audio-based music classification and annotation. *IEEE transactions on multimedia*, 13(2): 303–319, 2011.

Gers, Felix A, Schmidhuber, Jürgen, and Cummins, Fred. Learning to forget: Continual prediction with lstm. 1999.

Ghosh, Shaona and Kristensson, Per Ola. Neural networks for text correction and completion in keyboard decoding. *arXiv preprint arXiv:1709.06429*, 2017.

Guyon, Isabelle, Weston, Jason, Barnhill, Stephen, and Vapnik, Vladimir. Gene selection for cancer classification using support vector machines. *Machine learning*, 46 (1-3):389–422, 2002.

Hastie, Trevor, Rosset, Saharon, Zhu, Ji, and Zou, Hui. Multi-class adaboost. *Statistics and its Interface*, 2(3): 349–360, 2009.

Howard, Sam, Silla Jr, Carlos N, and Johnson, Colin G. Automatic lyrics-based music genre classification in a multilingual setting. In *Proceeedings of the Thirteenth Brazilian Symposium on Computer Music*, 2011.

Jafarpour, Borna, Matwin, Stan, et al. Boosting text classification performance on sexist tweets by text augmentation and text generation using a combination of knowledge graphs. In *Proceedings of the 2nd Workshop on Abusive Language Online (ALW2)*, pp. 107–114, 2018.

Jin, Haifeng, Song, Qingquan, and Hu, Xia. Efficient neural architecture search with network morphism. *arXiv preprint arXiv:1806.10282*, 2018.

Joulin, Armand, Grave, Edouard, Bojanowski, Piotr, Douze, Matthijs, Jégou, Hérve, and Mikolov, Tomas. Fasttext.zip: Compressing text classification models. *arXiv preprint arXiv:1612.03651*, 2016.

LeCun, Yann, Bottou, Léon, Bengio, Yoshua, Haffner, Patrick, et al. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11): 2278–2324, 1998.

Lee, Jin Ha and Downie, J Stephen. Survey of music information needs, uses, and seeking behaviours: Preliminary findings. In *ISMIR*, volume 2004, pp. 5th. Citeseer, 2004.

Lemaître, Guillaume, Nogueira, Fernando, and Aridas, Christos K. Imbalanced-learn: A python toolbox to tackle the curse of imbalanced datasets in machine learning. *Journal of Machine Learning Research*, 18(17):1–5, 2017. URL http://jmlr.org/papers/v18/16-365.

Libeks, Janis and Turnbull, Douglas. You can judge an artist by an album cover: Using images for music annotation. *IEEE MultiMedia*, 18(4):30–37, 2011.

Lippens, Stefaan, Martens, Jean-Pierre, and De Mulder, Tom. A comparison of human and automatic musical genre classification. In *2004 IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 4, pp. iv–iv. IEEE, 2004.

Luštrek, Mitja. Overview of automatic genre identification. *Ljubljana, Slovenia: Jožef Stefan Institute, Department of Intelligent Systems*, 2006.

Mayer, Rudolf and Rauber, Andreas. Multimodal aspects of music retrieval: Audio, song lyrics–and beyond? In *Advances in music information retrieval*, pp. 333–363. Springer, 2010.

Mayer, Rudolf and Rauber, Andreas. Musical genre classification by ensembles of audio and lyrics features. In *Proceedings of International Conference on Music Information Retrieval*, pp. 675–680, 2011.

Mayer, Rudolf, Neumayer, Robert, and Rauber, Andreas. Rhyme and style features for musical genre classification by song lyrics. In *Ismir*, pp. 337–342, 2008.

McKay, Cory and Fujinaga, Ichiro. Musical genre classification: Is it worth pursuing and how can it be improved? In *ISMIR*, pp. 101–106, 2006.

MetroLyrics. Song lyrics | metrolyrics. URL http://www.metrolyrics.com/.

Mikolov, Tomas, Sutskever, Ilya, Chen, Kai, Corrado, Greg S, and Dean, Jeff. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pp. 3111–3119, 2013.

Mishra, Gyanendra. 380,000 lyrics from metrolyrics, Jan 2017. URL https://www.kaggle.com/gyani95/380000-lyrics-from-metrolyrics.

MusixMatch. Explore the world's largest catalog of song lyrics and translations. URL https://www.musixmatch.com/. Online; accessed 13-February-2019.

Neumayer, Robert and Rauber, Andreas. Multi-modal music information retrieval: visualisation and evaluation of clusterings by both audio and lyrics. In *Large Scale Semantic Access to Content (Text, Image, Video, and Sound)*, pp. 70–89. LE CENTRE DE HAUTES ETUDES INTERNATIONALES D'INFORMATIQUE DOCUMENTAIRE, 2007.

Norvig, Peter. How to write a spelling corrector, Feb 2007. URL https://norvig.com/spell-correct.html.

Nyberg, Adam. Classifying movie genres by analyzing text reviews. *arXiv preprint arXiv:1802.05322*, 2018.

Oramas, Sergio, Nieto, Oriol, Barbieri, Francesco, and Serra, Xavier. Multi-label music genre classification from audio, text, and images using deep features. *arXiv preprint arXiv:1707.04916*, 2017.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

Pennington, Jeffrey, Socher, Richard, and Manning, Christopher D. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1532–1543, 2014. URL http://www.aclweb.org/anthology/D14-1162.

Powers, D.M.W. Evaluation: from precision, recall and f-measure to roc, informedness, markedness and correlation. *Journal of Machine Learning Technologies*, 2(1): 37–63, 2011.

Provost, Foster and Fawcett, Tom. Robust classification for imprecise environments. *Machine learning*, 42(3): 203–231, 2001.

Ramos, Juan et al. Using tf-idf to determine word relevance in document queries. In *Proceedings of the first instructional conference on machine learning*, volume 242, pp. 133–142, 2003.

Schedl, Markus, Zamani, Hamed, Chen, Ching-Wei, Deldjoo, Yashar, and Elahi, Mehdi. Current challenges and visions in music recommender systems research. *International Journal of Multimedia Information Retrieval*, 7 (2):95–116, 2018.

Sturm, Bob L. A survey of evaluation in music genre recognition. In *International Workshop on Adaptive Multimedia Retrieval*, pp. 29–66. Springer, 2012.

Sturm, Bob L. Classification accuracy is not enough. *Journal of Intelligent Information Systems*, 41(3):371–406, Dec 2013. ISSN 1573-7675. doi: 10.1007/s10844-013-0250-y. URL https://doi.org/10.1007/s10844-013-0250-y.

Tsaptsinos, Alexandros. Lyrics-based music genre classification using a hierarchical attention network. *arXiv preprint arXiv:1707.04678*, 2017. Online; accessed 13-February-2019.

Valverde-Albacete, Francisco J and Peláez-Moreno, Carmen. 100% classification accuracy considered harmful: The normalized information transfer factor explains the accuracy paradox. *PloS one*, 9(1):e84217, 2014.

Wang, Baoxin. Disconnected recurrent neural networks for text categorization. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pp. 2311–2320, 2018.

Yang, Zichao, Yang, Diyi, Dyer, Chris, He, Xiaodong, Smola, Alex, and Hovy, Eduard. Hierarchical attention networks for document classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 1480–1489, 2016.

Ying, Teh Chao, Doraisamy, Shyamala, and Abdullah, Lili Nurliyana. Genre and mood classification using lyric features. In *2012 International Conference on Information Retrieval & Knowledge Management*, pp. 260–263. IEEE, 2012.

Zoph, Barret, Vasudevan, Vijay, Shlens, Jonathon, and Le, Quoc V. Learning transferable architectures for scalable image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 8697–8710, 2018.

8. Appendix I

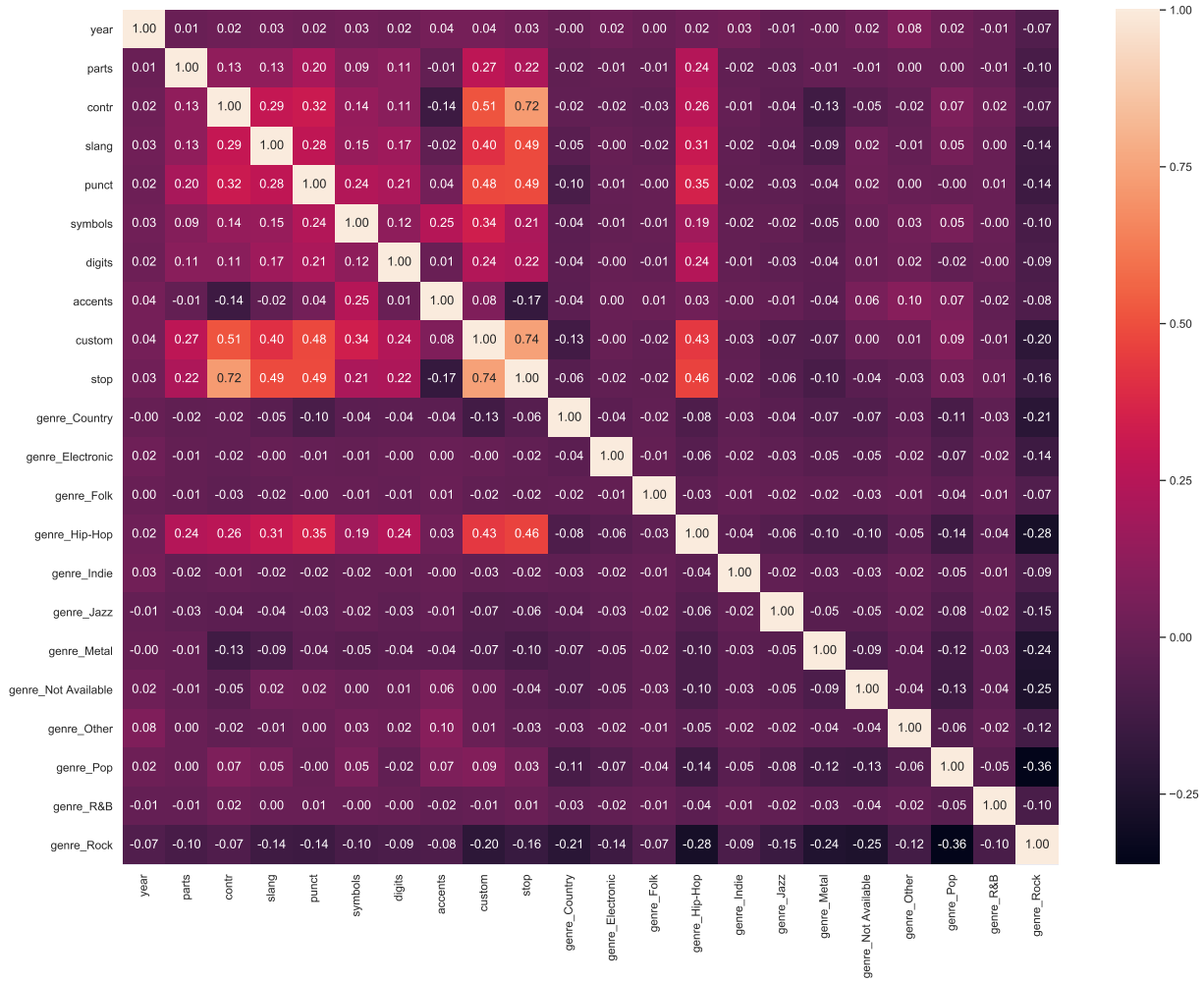

Figure 6. Correlation Matrix

9. Appendix II

```
                    Results of Test Set
                    --------------------
Accuracy: 0.6327179106554716, F1 Score: 0.40741835691818495,
Precision: 0.5561818540440588, Recall: 0.367799810387632

              precision    recall  f1-score   support

           0       0.54      0.36      0.43      2795
           1       0.42      0.13      0.20      1321
           2       0.53      0.12      0.19       355
           3       0.83      0.81      0.82      4700
           4       0.36      0.03      0.05       585
           5       0.52      0.30      0.38      1504
           6       0.64      0.54      0.59      3669
           7       0.52      0.43      0.47      7506
           8       0.56      0.14      0.22       666
           9       0.64      0.82      0.72     20013

   micro avg       0.63      0.63      0.63     43114
   macro avg       0.56      0.37      0.41     43114
weighted avg       0.62      0.63      0.61     43114
 samples avg       0.63      0.63      0.63     43114


[[ 1002     3     3    14     2    81    17   233     2  1438]
 [   13   175     0    73     1     9    56   236     2   756]
 [   23     3    42     2     1     7    15    52     0   210]
 [   13    29     2  3808     1     4    37   335     3   468]
 [   10     3     1     9    16     3    11    47     0   485]
 [   92    14     1    21     1   453    11   207    14   690]
 [    6    25     3    72     1     5  1990   122     3  1442]
 [  164    69     7   306    10    99   121  3218    18  3494]
 [    9     1     0    20     0    19     8   134    90   385]
 [  510    97    20   257    12   191   854  1559    28 16485]]
```

Figure 7. Results of best performing model on test set