Information Retrieval and Text Mining

Project 2

Author: Cojocariu Sebastian

Group: 507 IA

Table of Contents

Introduction

For this project, we had at our disposal a set of lyrics and their genre, divided into train/test sets. The task was to design several models that could predict the latter based solely on the raw text.

The data consists of 18513 entries for the train set, and 7935 for the test (hold-out) set. First, we removed any row that has a null entry, and then filtered based on empty lyrics (or trailing spaces). These dataset were then shuffled. There are a total of 10 Genres, their frequency being summarised in Figure 1. As usual, during the trainings, each genre was then mapped to an integer, based on their frequencies's descending order. Additionally, a sanitisation method was designed to remove non-characters (e.g. non-ascii, punctuation, trailing whitespaces).

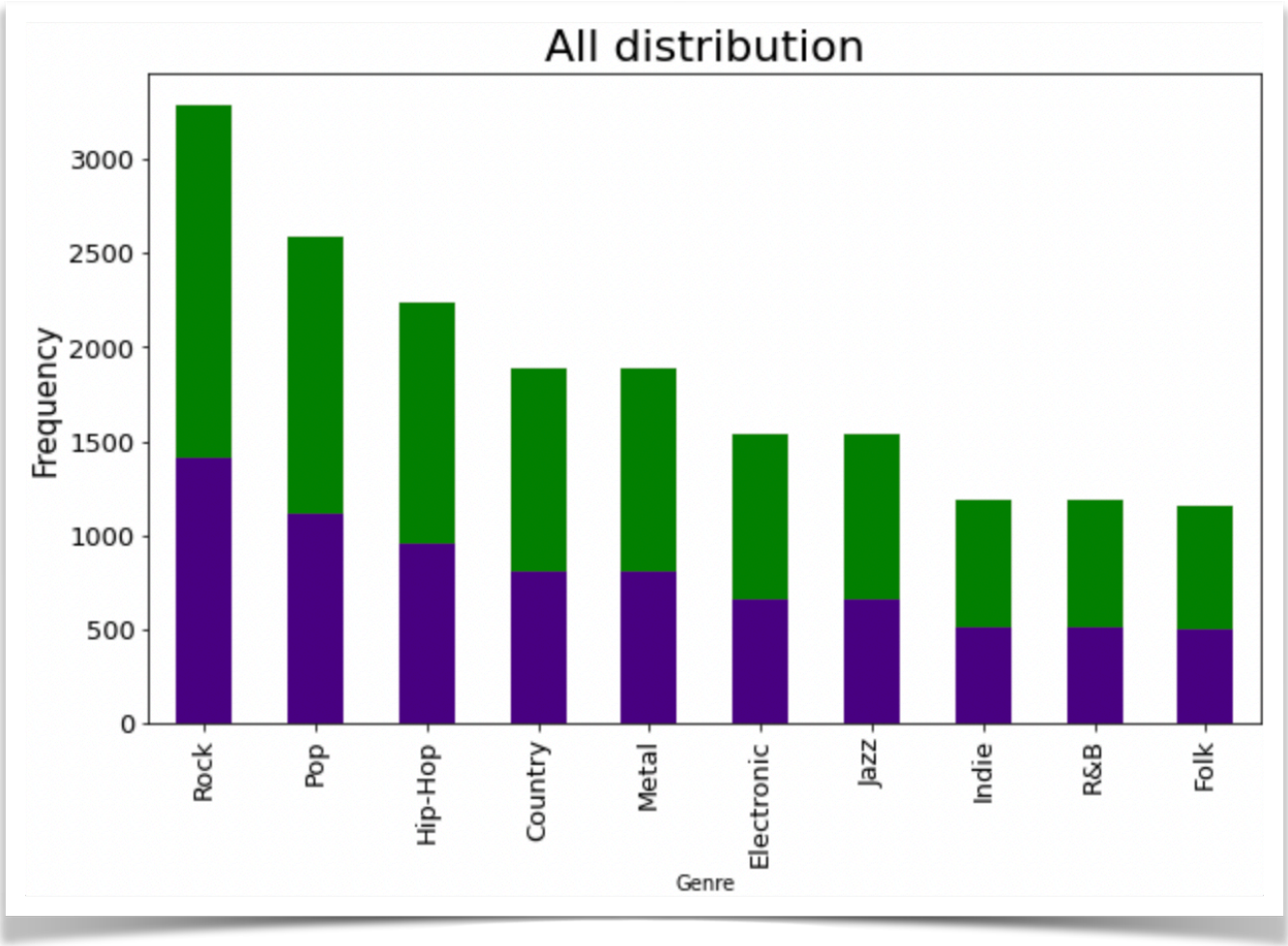

Figure 1. Genre distribution (green for train set, indigo for test set respectively).

For each genre, we then inspected the most frequent words that appear in their poems. An immediate observation was that were lots of words (e.g. love, know, like) that overlapped between different genres, which pointed at the difficulty of the task (Figure 2).

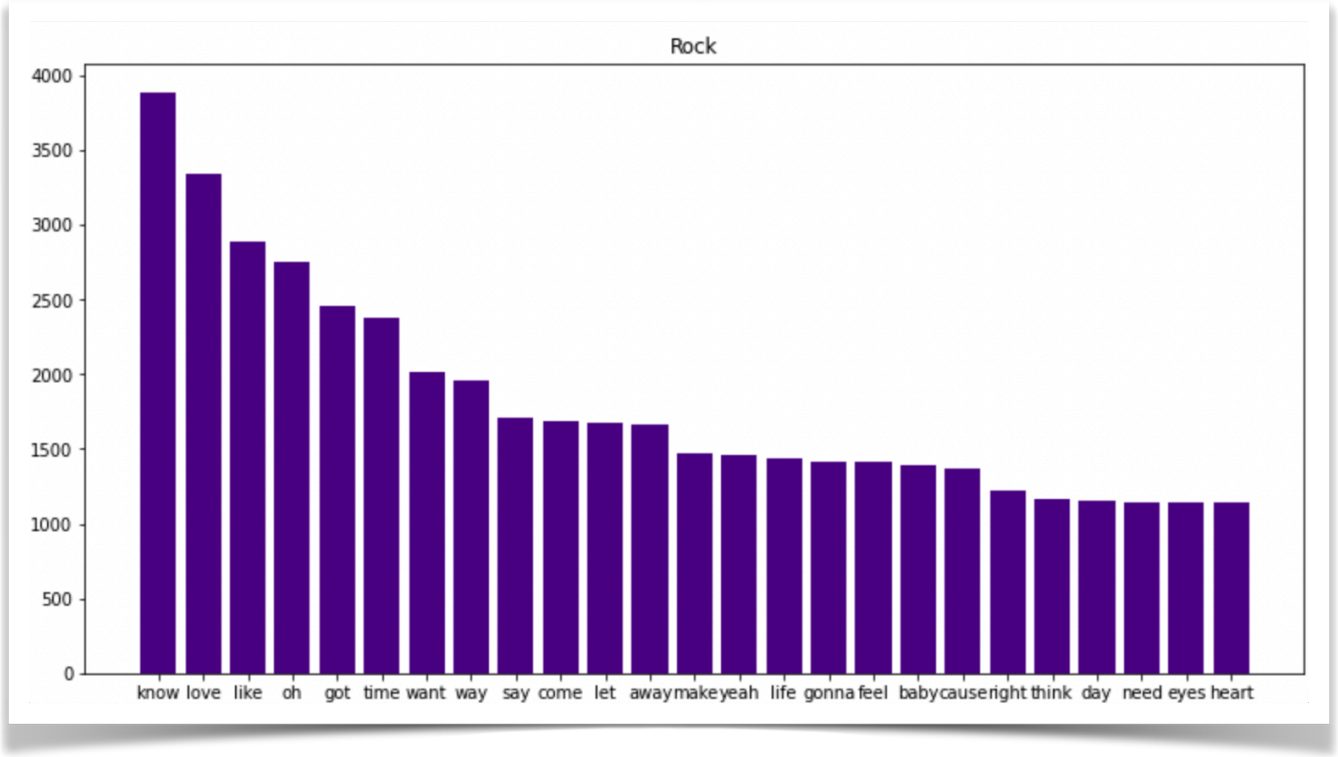

Figure 2. The top words for the Rock genre (their frequency).

Our initial approach was to create some features that could be useful to predict genres against. For this, we compute multiple types:

Musical Features

Rhythm and Rhyme

- Average / Standard deviation / Minimum / Maximum / Percentiles 25-50-75 for the number of words per lines.
- Average / Standard deviation / Minimum / Maximum / Percentiles 25-50-75 for the number of syllables per lines.
- Average / Standard deviation / Minimum / Maximum / Percentiles 25-50-75 for the number of letters per line.
- Average / Standard deviation / Minimum / Maximum / Percentiles 25-50-75 for the number of rhymes per blocks (of length 8).
- The number of rhymes + the number of rhymes normalized (taking into account all possible combinations of rhymes across all the poems).

Semantical features

- The number of stopwords + the number of stopwords normalized (taking into account the total number of words).
- The frequency of various part of speeches, as per NLTK (e.g. verb, adjective, noun).

General metrics

- The number of lines the poem has.
- The number of words the poem has.
- The number of letters the poem has.

Readability, Complexity, and Grade level

- Flesch Reading Ease.
- Flesch Kincaid Grade.
- Smog Index.
- Coleman Liau Index.
- Automated Readability Index.
- Dale Chall Readability Score.
- Difficult Words.
- Linsear Write Formula.
- Gunning Fog.
- Fernandez Huerta.
- Szigriszt Pazos.
- Gutierrez Polini.
- Crawford.
- Gulpease Index.
- Osman.

The total of features extracted per text was 62. A visualisation of the features is shown in Figure 3, using PCA Decomposition. We can easily observe that the data overlaps too much and that there is no exact separability between classes (at least not converting them into a 2-dimension representation).

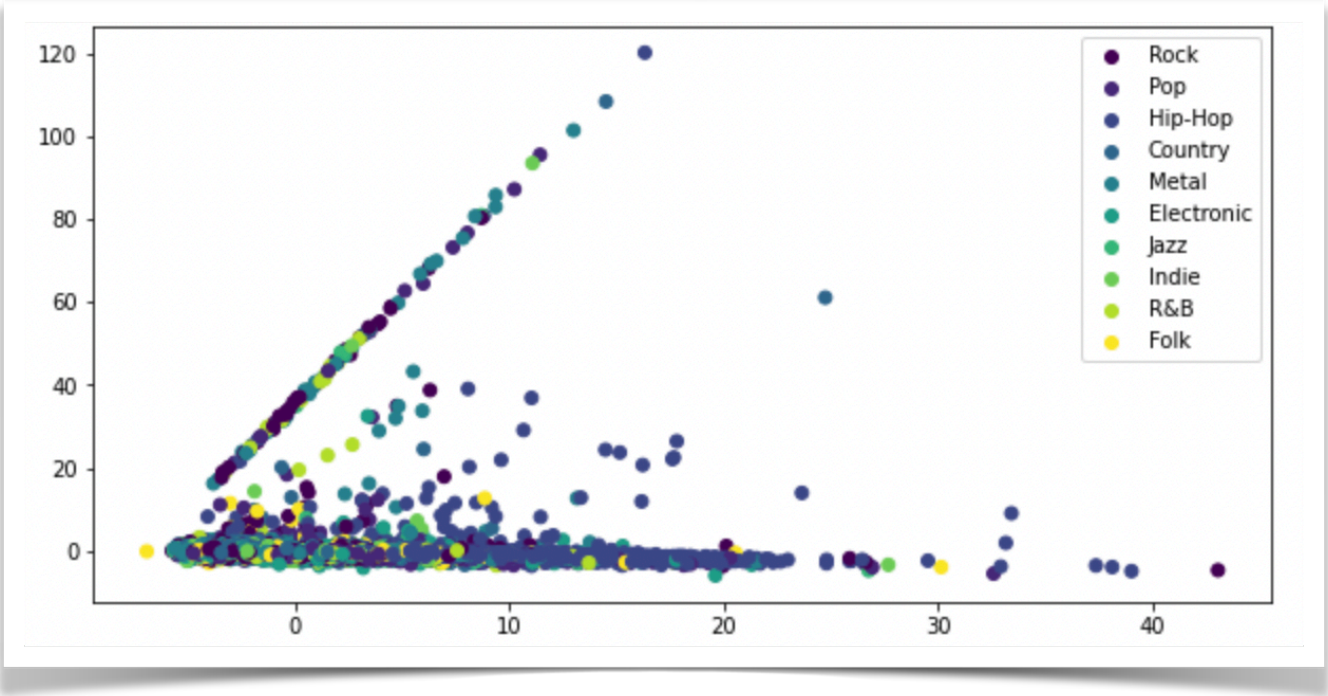

Figure 3 - Musical Features Visualisation.

BOW Features

For this features, we used a TF-IDF approach using the most frequent 1000 words. We also tested with ngrams in the range (3,5) but the results were not improved by much. Increasing this to 20k would have generated a better performance by ~1.5%, but we wanted to use these features in conjuction with other types of features, so we imposed a reasonable limit for this. As for the data vizualization, we can see that the data is a little bit better representated in clusters (although different genres still overlap) (Figure 4). For instance, *Electronic* and *Jazz* are almost separated by *Hip-hop* and *Pop*, while *Rock* seems to share features on both clusters.

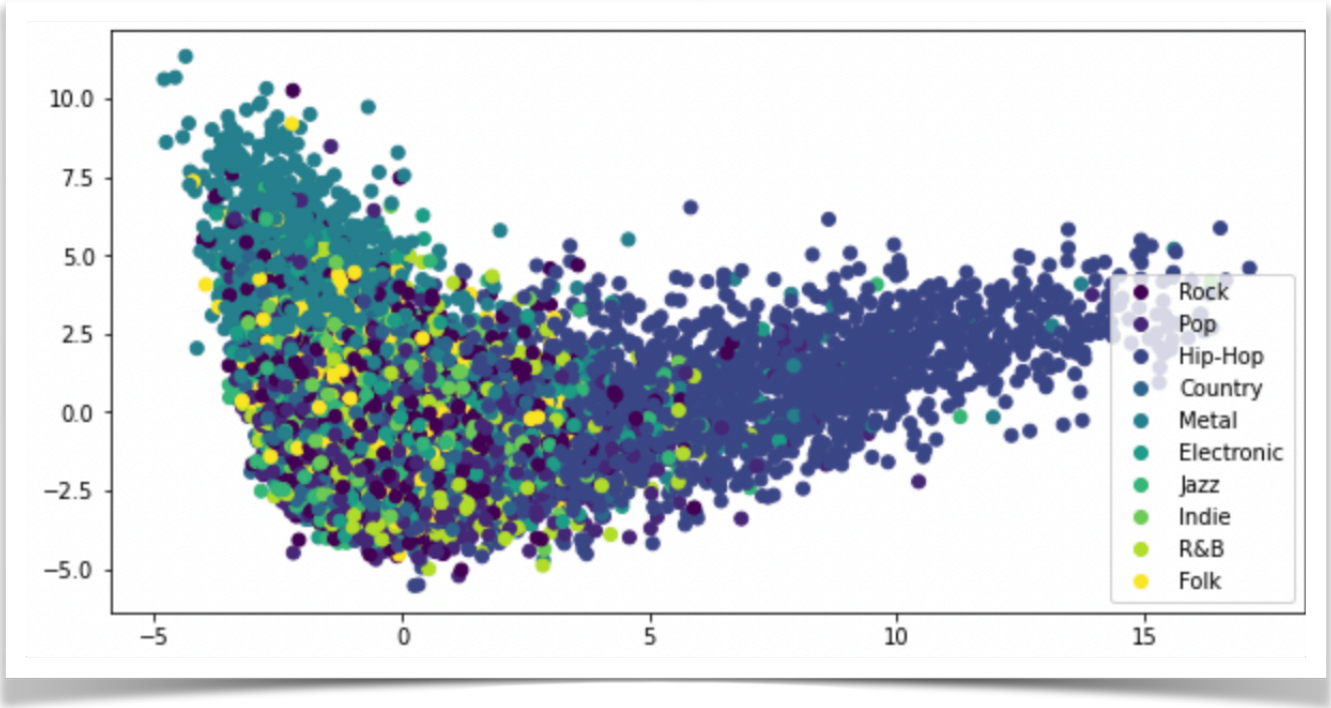

Figure 4 - BOW Features Visualisation.

Dense Representation Features using Transformers

We used `all-MiniLM-L6-v2` pertained model provided by sentence-transformer package to encode text into a 384 dense representation. These features were then used to train several models. The data visualisation using this method seem to do a clearly better job in distinguishing among genres than before (although there are still overlappings since a topic might be used in different genres, although arguably in *smaller* quantities) (Figure 5).

Sentence Embedding Features using FastText word embeddings

To generate sentence embeddings, we averaged all the existent word embeddings (given by *FastText*) existent in the lyrics. Stop-words were removed as they usually do not offer much contextual information. The total number of features per sample is 300. The results in data visualisation seem more noisy than before, with lots of overlappings. This was expected, as by

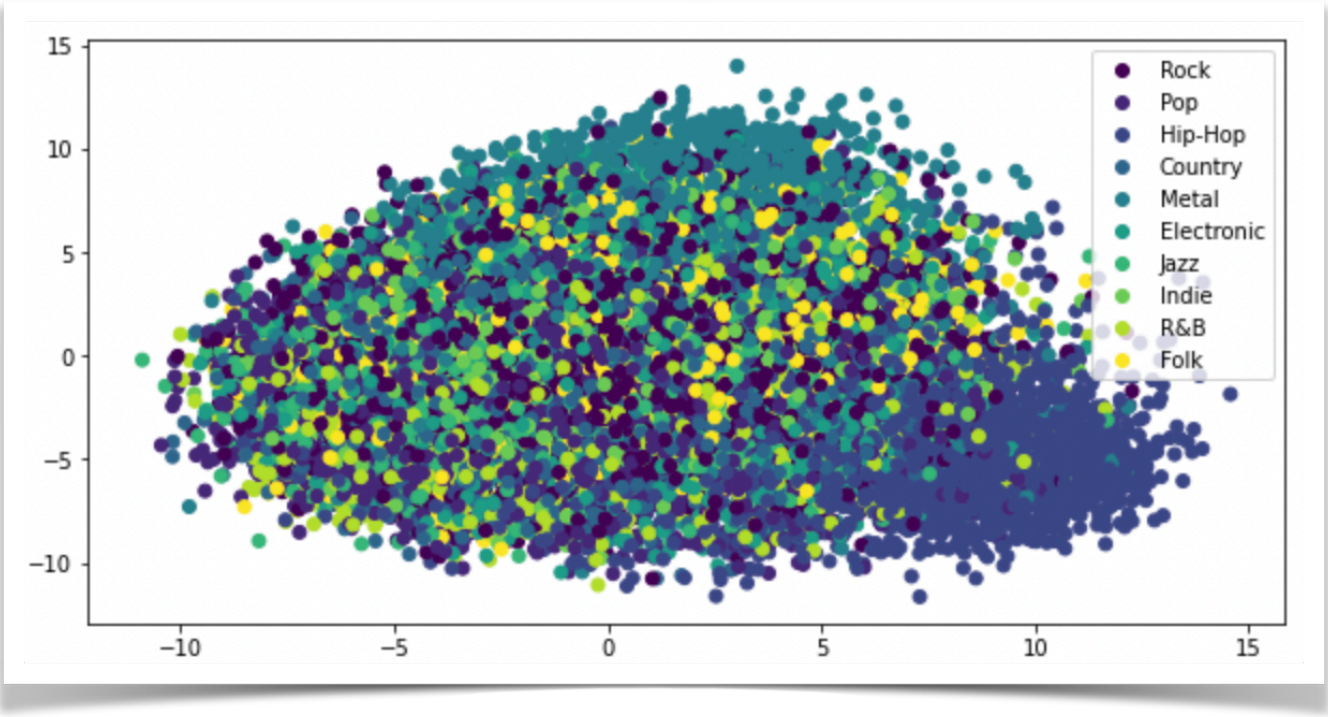

Figure 5 - Dense Representation Features Visualisation.

averaging we lose a lot of contextual information (relative order of words and how they interact with each other, since the embeddings are static, and not dynamically computed as they were before by using a transformer architecture). Looking at the data, it seems that there is a lot of noise (Figure 6).

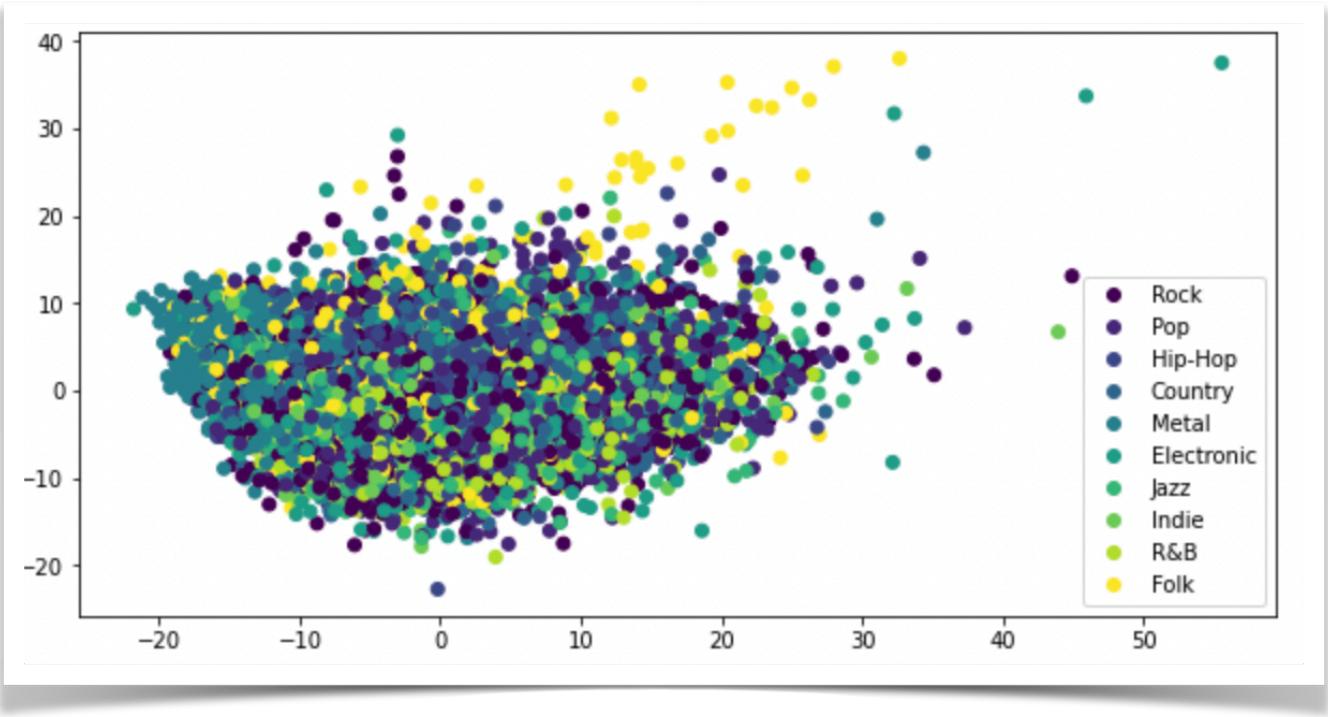

Figure 6 - FastText Features Visualisation.

Aggregate multiple Features

We aggregated musical, BOW, dense representation (not the embeddings from FastText though as they are somehow "included" in a way in the representation given by the transformer). The BOW features are used to offer an intuition of the most relevant words per poem (the transformer can offer an idea about the what the poem is about, but not really about which words were used, which might provide some extra information). The total number of features per sample is 1446. They way data look may suggest a technique involving a hierarchical approach in modelling (Figure 7).

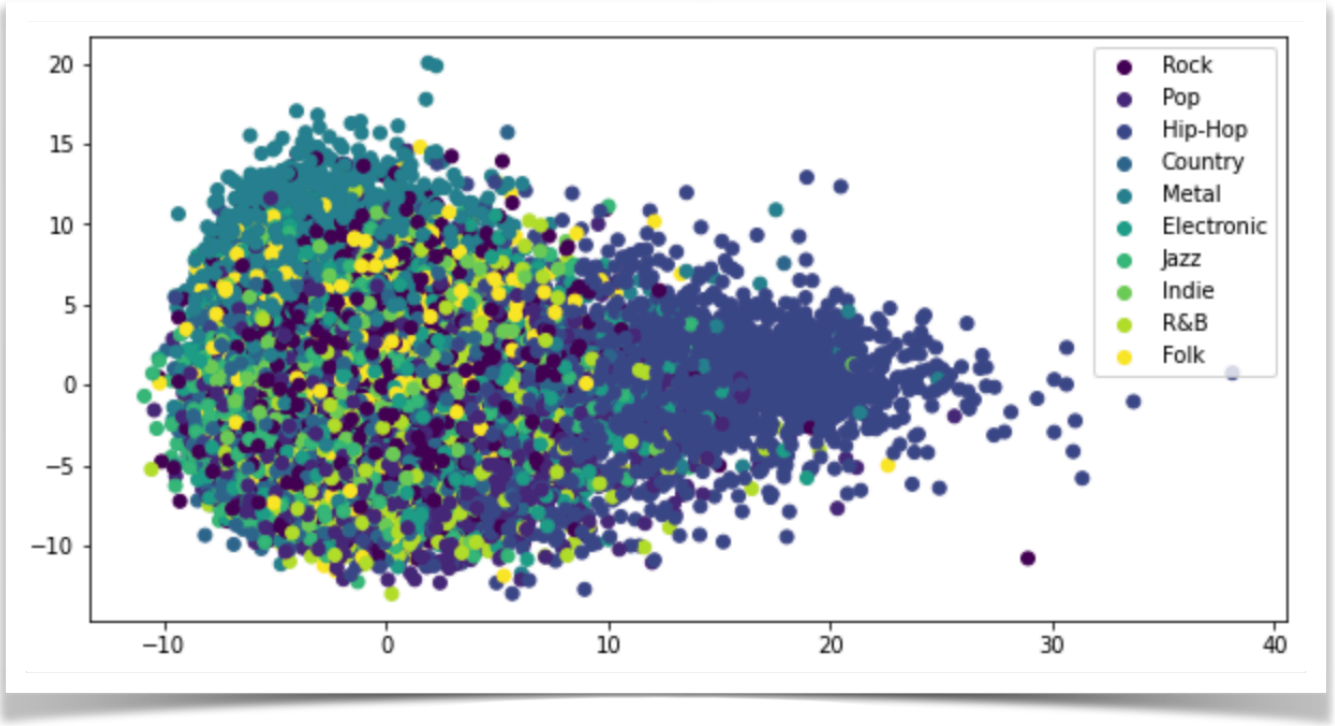

Figure 7 - Aggregate Features Visualisation.

Modeling

For each of the model that was testes, we did a 3-fold CV. Then, the average of these models against the test (hold-out) set recorded. The train set was split in a 85-15 fashion.

Non-learner models

1. **Pure Random:** randomly predict a genre.

2. **Weighted Random**: randomly sample a label, but each genre has a weight associated with it (more frequent genre will be sampled more).

3. **Most Frequent:** predicting the most frequent genre.

Easily, the method of predicting the most frequent class from the training set achieve the best score among the random sampling methods. Our baseline model will be better than **17.77**% as we will see in the incoming sections (Figure 8).

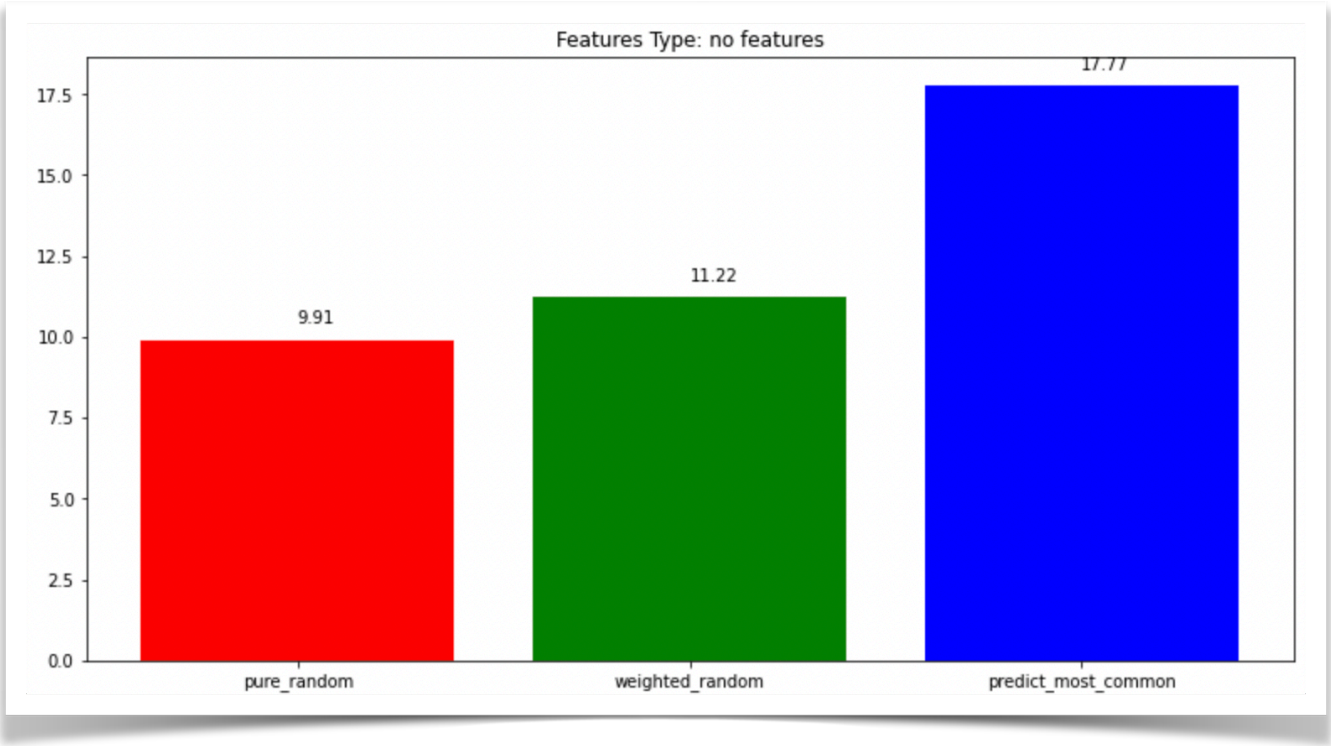

Figure 8. Non-learning models.

Shallow models

1. Using Musical Features

The results indicate that the performance obtained is at **34.3%** (with slight modification in prediction among the four tested models), almost double the previous "method" accuracy (predicting the most frequent class). The model architecture seems to not statistically affect performance (Figure 9).

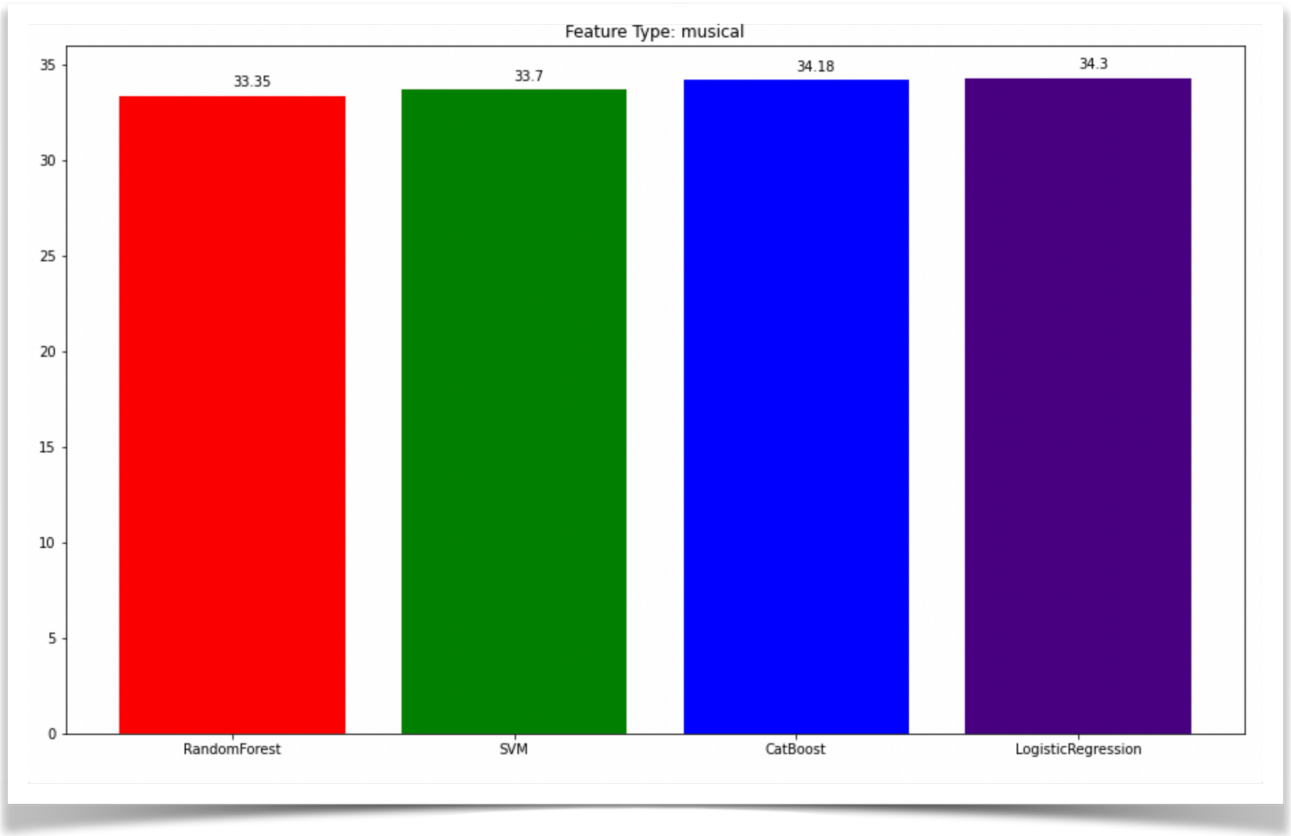

Figure 9 - Musical Features performance against four models.

2. Bag Of Words

The models' performance increase from the previous experiment, the best (CatBoost) reaching a staggering **40%**. This behaviour suggests that there are some words that are representative for a set of genres collection. We will use this features for aggregation to see if adding contextual features (from transformer) + musical features improves performance (Figure 10).

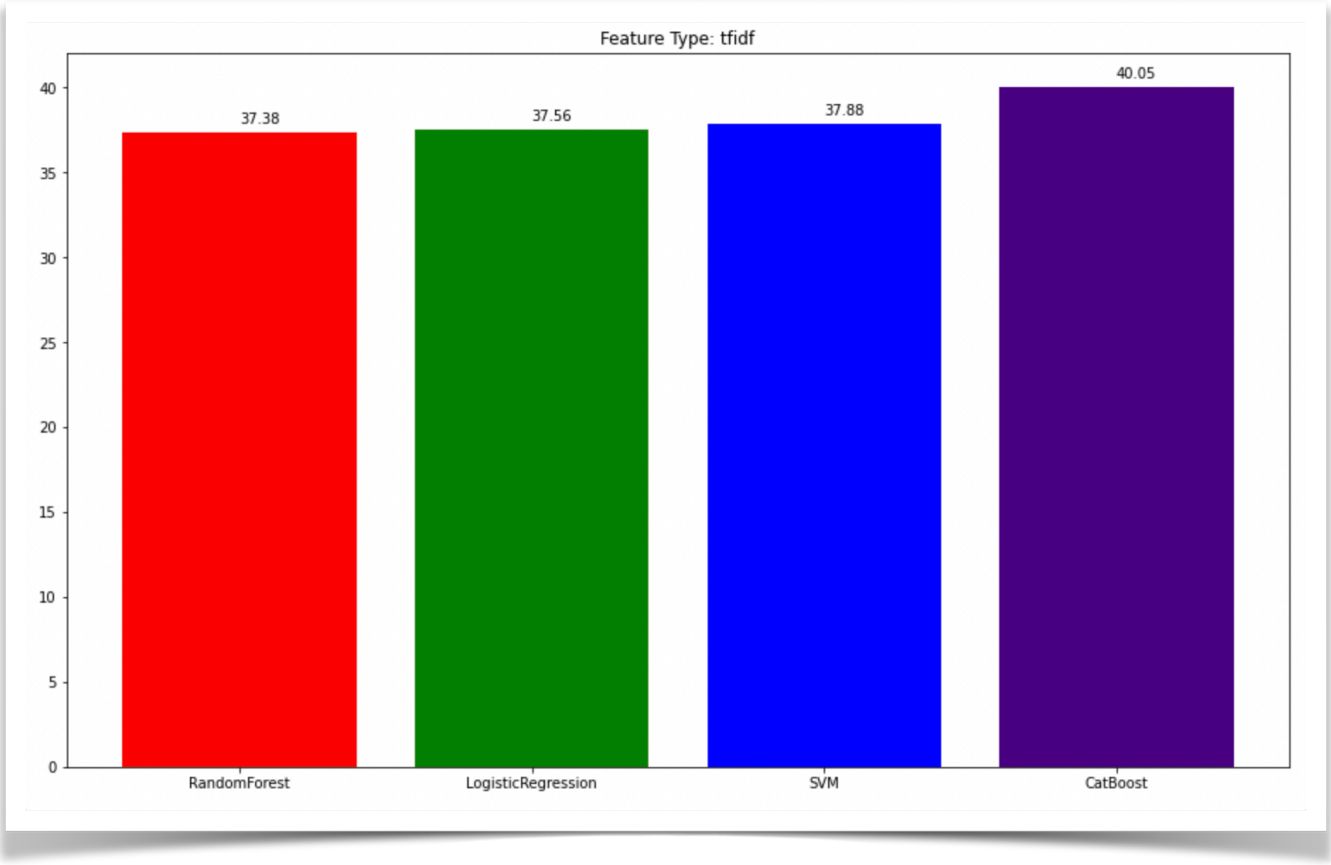

Figure 10 - BOW Features performance against four models.

3. Sentence Embedding from Transformer.

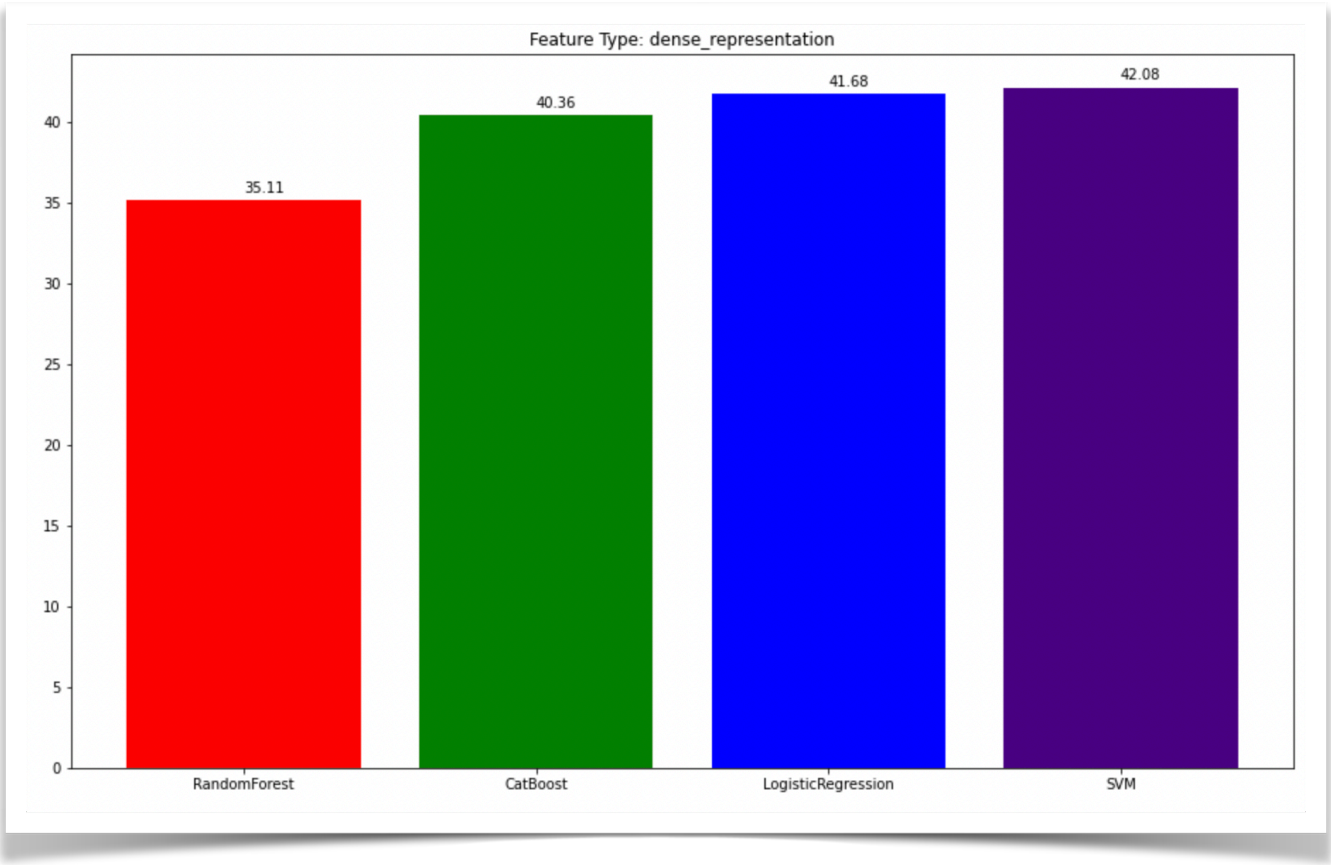

Figure 11 - Dense Representation Features performance against four models.

For this experiment the results are further improving, the best model (SVM) reaching about **42%** accuracy, followed closely by Logistic Regression and CatBoost, each with a performance of over **40%** (Figure 11). These results alone suggest that the embedding provided by the transformer architecture manage to better capture the meaning of the lyrics and thus, offering better prediction results overall.

4. Sentence Embedding from FastText

The results using as sentence embedding the averaging of each word's embedding (given by *FastText*) offer poorer results (around **38.1**%), although the *CatBoost* model did manage to beat the best model trained with musical features (Figure 12). This might suggest that the musical features cannot exhaustively capture all the genres' traits, and so they are not suitable enough to predict only against them. In the next experiment we will try to aggregate features from multiple sources (*musical + sentence embedding from transformer + BOW*) and check if this solution improves the end-to-end performance.

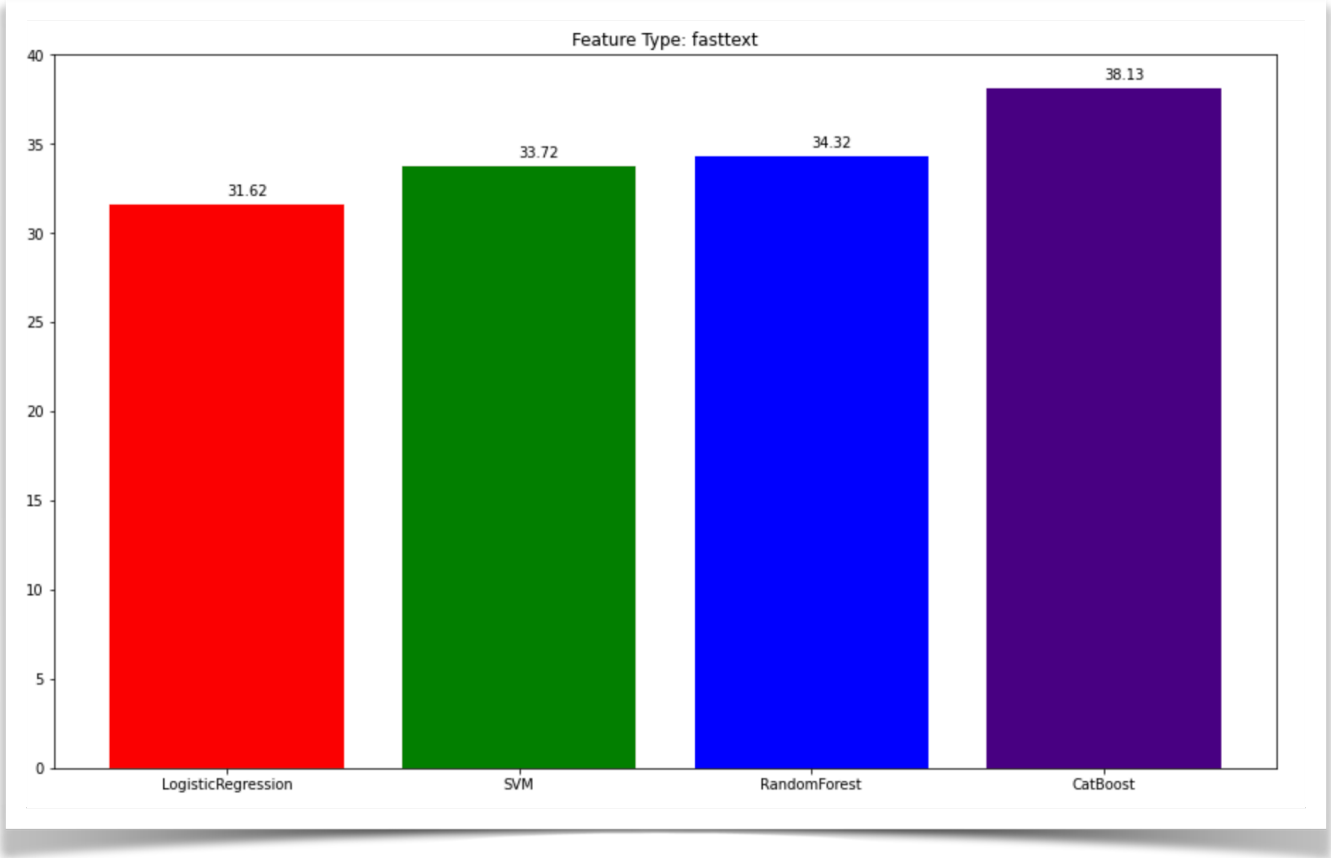

Figure 12 - FastText Features performance against four models.

5. All features combined (Musical + Dense Representation + BOW)

The results exhibited in this section show that aggregating musical, contextual and BOW features indeed increased the end-to-end performance of our models. Both *LogisticRegression* and *CatBoost* reach a staggering **43.5%** accuracy performance. The other two models (*SVM* and *RandomForest*) managed to obtained around **38%** (Figure 13). Considering that we did not properly tuned the parameters of these models, we can safely assume that we could actually increase these numbers by **1-2%**, which would provide an astonishing performance.

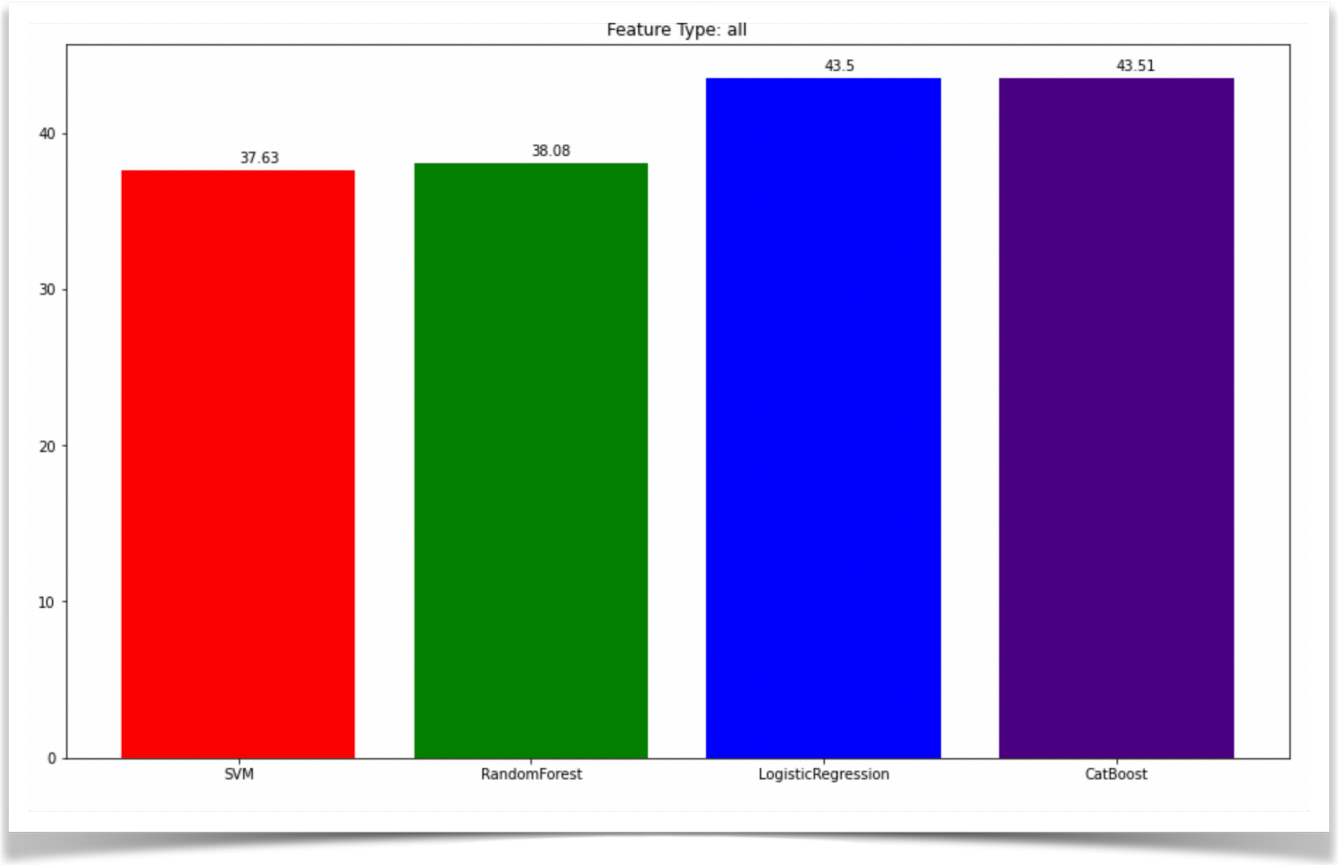

Figure 13 - Aggregate Features performance against four models.

Additionally, we chose the best model (Logistic Regression trained on the aggregated features) and performed a hyper parameter tuning using a GridSearch Technique, reporting the results per genre in Figure 14 (confusion matrix). More, the this fine-tuned model achieved a **44.57**%, with **1**% more.

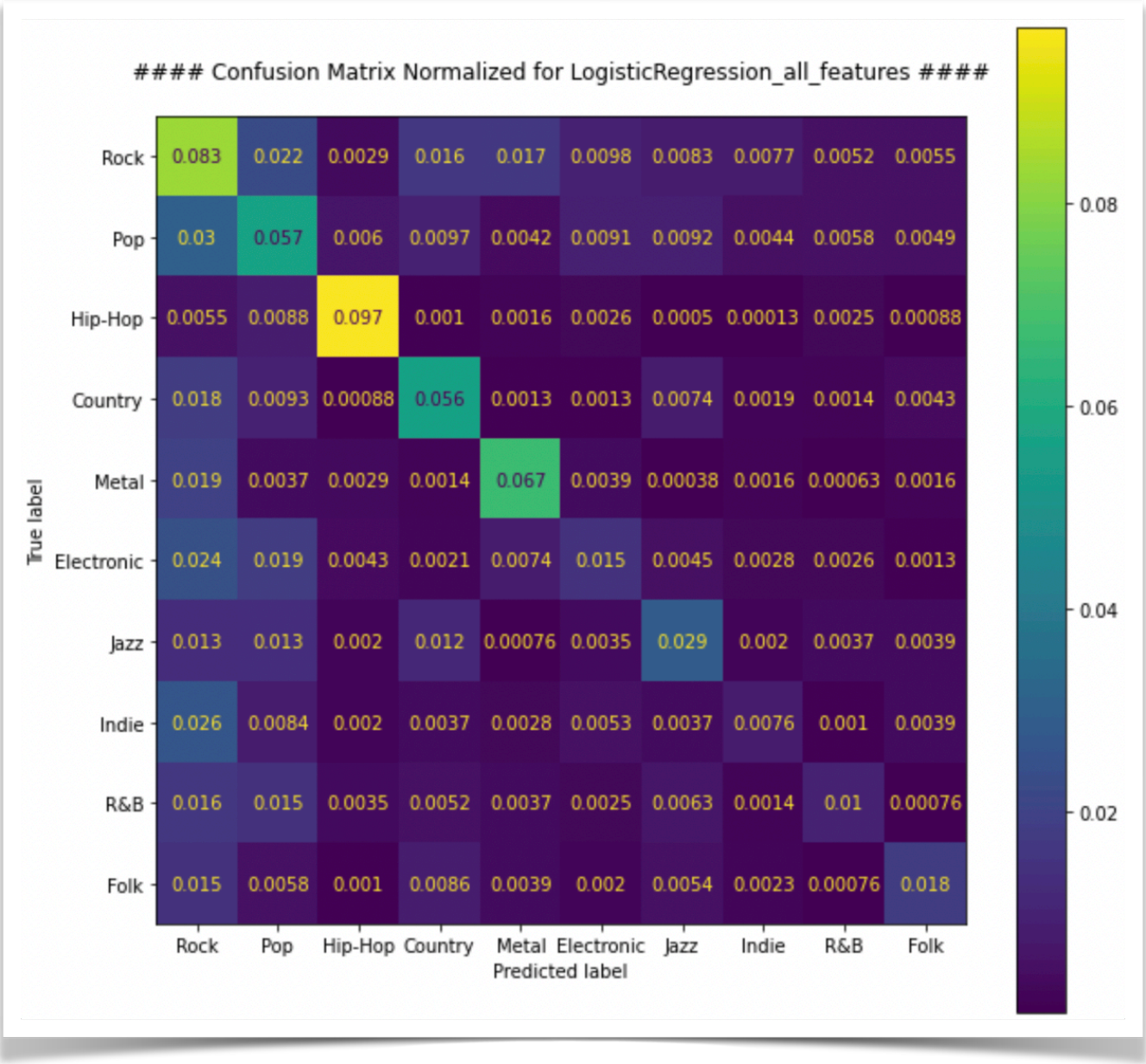

Figure 14 - Confusion Matrix of the best model (Logistic Regression).

Deep Learning Models

1. HybridCNN (Convolutional Character NN + Convolutional Word NN).

The architecture consists of two parts (which can be discarded during training):

- A character-level CNN
- A word-level CNN

The first part is described perfectly here. The vocabulary of characters is selected based on the frequency of these characters (the most rare are discarded to reduce vocabulary size). More specifically, for a character to be considered part of the vocabulary, the number of its occurrences must be bigger than a minimum threshold (customisable).

The idea to consider the word-level CNN arise from the fact that maybe position of multiple words combined can be relevant to create useful features (the order of words can be important in some regions). In order to reduce the dimension of the vocabulary, each word was mapped to its stemmed conversion in the dictionary (meaning that the dictionary contains only stemmed words as keys). Also, for a word to be considered part of the vocabulary, the number of its occurrences must be bigger than a minimum threshold (which is again customisable).

Accuracy on test set: **35.2%**

2. Transformer Architecture

We made use of a pretained model from huggingface called bert-base-uncased, which we fine-tuned accordingly to our classification use-case. This model introduces a new mechanism in Natural Language Understanding called attention block. More specifically, it consists of a bidirectional auto-encoder (instead of the classical unidirectional encoder used in recurrent neural networks) that allows words to update their embeddings based on their relative position to one another (left and right), instead of using a statical embedding for each word (e.g. *FastText, Word2Vec, GloVe*).

Accuracy on test set: **43.8%**

Conclusions

- Four different types of features (*musical, sentence transformer, FastText and BOW*) were tried, plus an additional type that aggregates three of these features. The best ones were those that used sentence embeddings from transformers.

- The final results indicate that Logistic Regression (C=0.5) is the best model, with a performance of **45.09**%, followed by Bert fine-tuned at around **43.8**% (trained for just 8 epochs). Immediately after are the LR and CatBoost trained on all features with default parameters, with a performance of **43.5**% each). Nevertheless, the other approaches obtain very good results, especially the one using sentence embedding (around **42**% and go up to almost **44**% when used in conjunction with other features) (Figure 15).

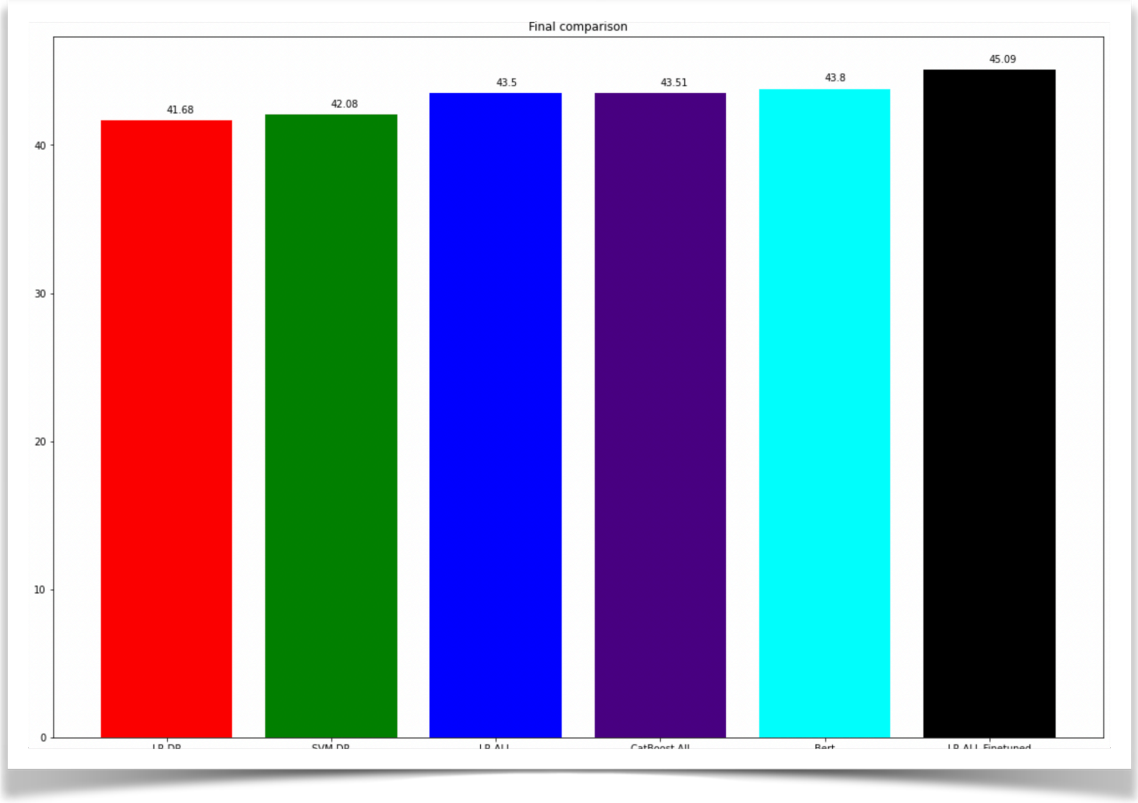

Figure 15 - Final Comparison

- Adding musical features on top of contextual ones (from transformer) plus BOW offers a boost in performance, which shows the importance of the rhythm and rhyme of the lyrics when predicting a musical genre.

- The *hybridCNN* did not managed to come close to the best trained models, but this may be due to not properly fine-tuning the model (it is known that Deep Learning models are more prone to easily overfit than shallow models when using default parameters).

- Even though the actual lyrics partially encode a melody, there are other things to be considered (the rhythm of the singer, the pronunciation, etc). This introduce a bias that is very hard to overcome just by simply looking at the raw text alone.

- As for training goes, we did tried both balanced *class_weights* as well as the default one, but the results proved to be better with the latter.

- For each experiment, we used a 3-fold cross validation (5 was used at times, but training 4 models for 5 types of features would have taken too much). The results seem to remain consistent though, regardless of the number of folds.

Future Agenda

- For each type of feature, choose the best model from CV and fine-tune its parameters.

- Implement a hierarchical model to first predict some clusters of genres, and internally to have other models that predict the final type (which might be different architectures for each cluster).

- Augment the data from other sources (although the requirement from the task did not allow us to do this).

- Use other metadata: for instance there are tools to extract audio features: e.g. *librosa* or check for precomputed features online (e.g. using *spotify, echonest*).

- Using the name of an Artist as well (since an artist usually account for only one genre, or maybe two). This can provide useful information about the text.