Information Retrieval and Text Mining

Project Documentation

Wali, Ahmad Mustapha (Gr. 507, Artificial Intelligence)

Project Summary

The project was aimed to predict a song's genre from only its lyrics. There were 2 files provided: a training dataset of 18513 samples, and a test dataset of 7935 samples. Each of the songs in both datasets belongs to one of 10 genres. The datasets did not have missing values, but the genre values are skewed. The test dataset was split into a validation set and a test set of equal sizes.

For the project, 4 classical machine learning models were trained and tested using different features and stylistic markups. The features used were bag of words (BoW), term frequency–inverse document frequency (TF-IDF), and Word2Vec; with a markup having stopwords and another excluding them.

Exploratory Data Analysis

The training dataset had 18513 rows and 7 columns, while the test dataset had 7935 rows and 7 columns. Only 2 columns (Lyrics and Genre) were used for the predictions. Both datasets did not have missing values. However, the label value-counts were skewed in both datasets, as they both have the same distributions. Rock had the highest percentage with approx. 18%, while folk had the lowest with approx. 7%.

A quick glance at the most frequent words in both the training and testing datasets shows that the most frequent words were all stopwords. "The" was the most frequent word with more than 163,000 occurrences in the training dataset and almost 70,000 occurrences in the test dataset.

Song	0
Song year	0
Artist	0
Genre	0
Lyrics	0
Track_id	0
labels	0

Table 1 Training Dataset Missing Values

Song	0
Song year	0
Artist	0
Genre	0
Lyrics	0
Track_id	0
labels	0

Table2 Testing Dataset Missing Values

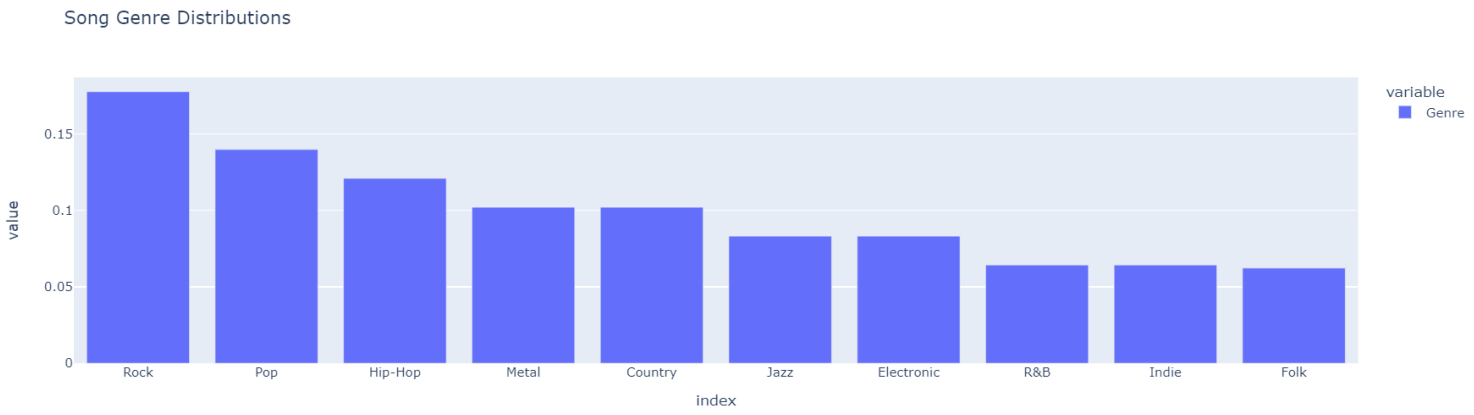

Figure 1 Song Genre Distribution for Trainset

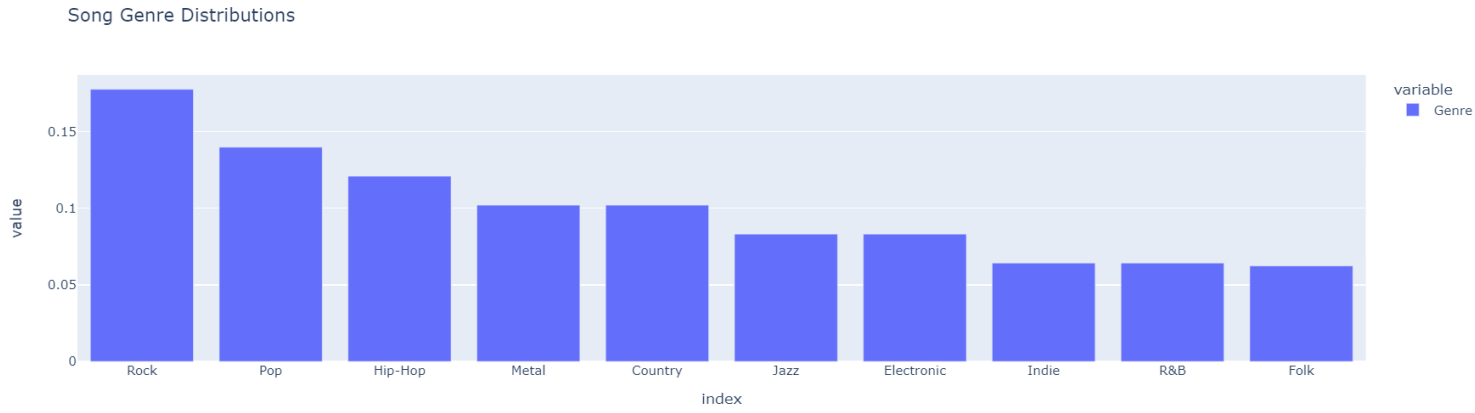

Figure 2 Song Genre Distribution for Testset

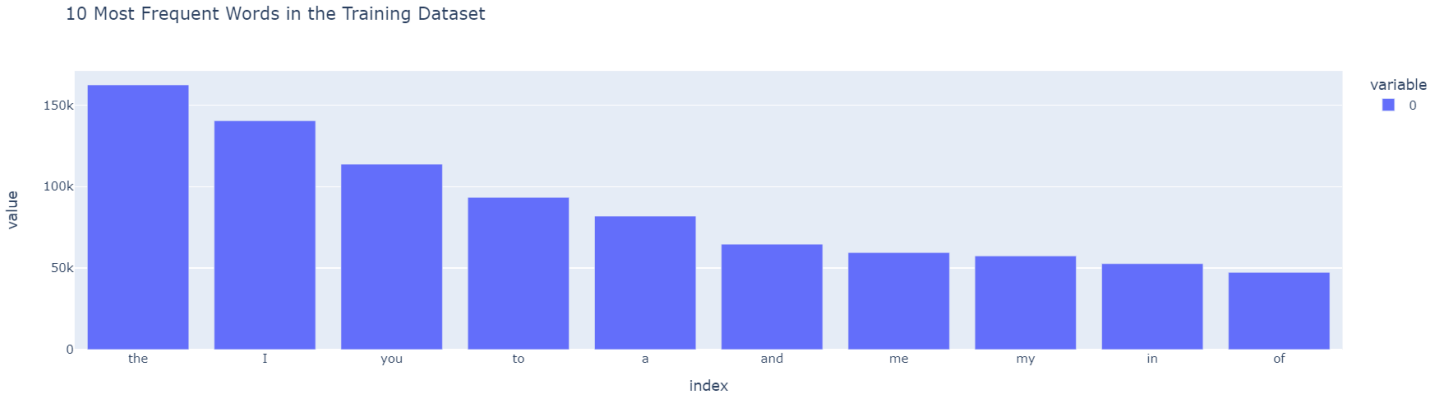

Figure 3 Most Frequent Words in Training Dataset

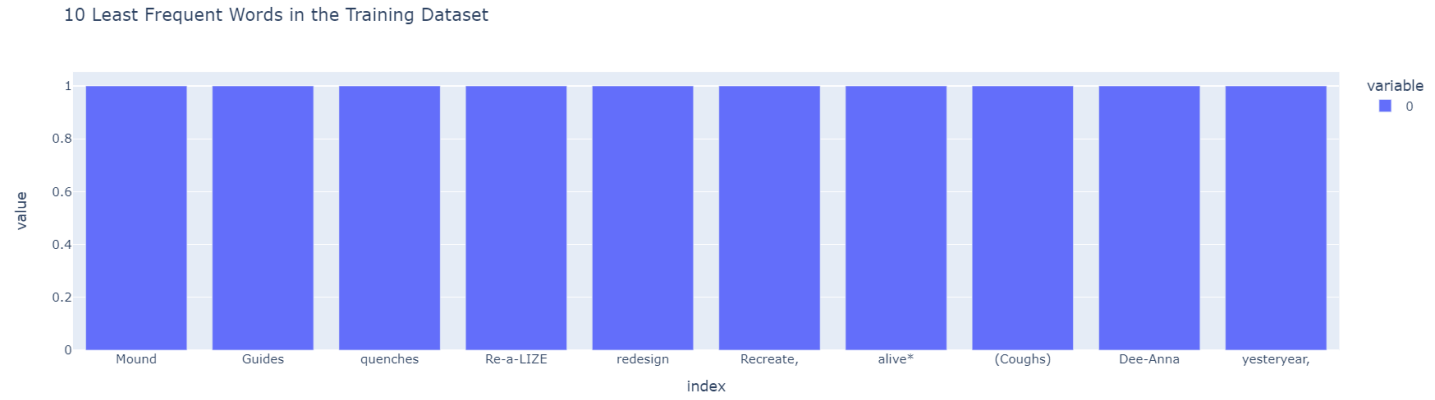

Figure 4 Least Frequent Words in Training Dataset

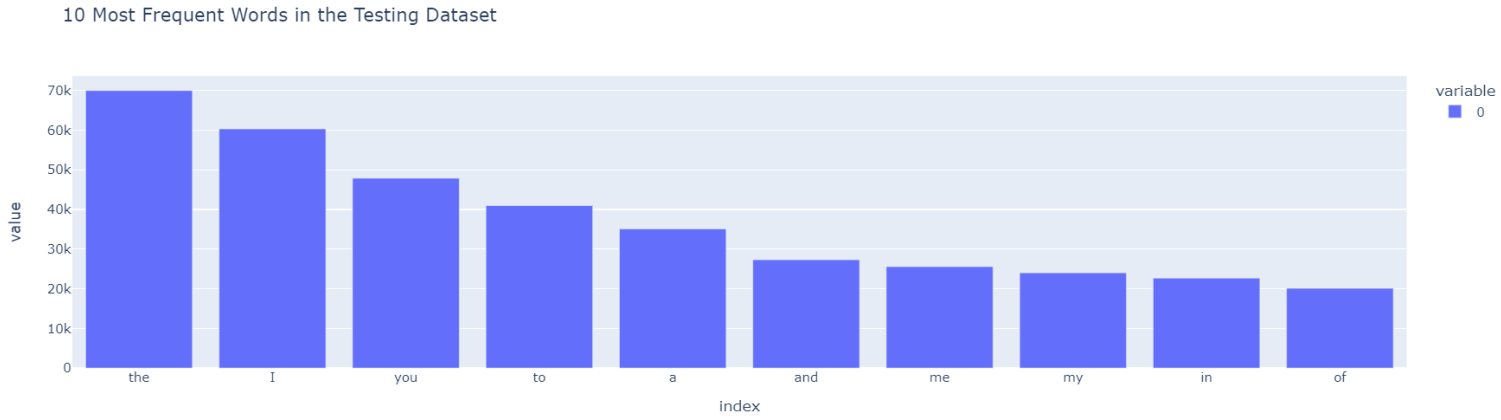

Figure 5 Most Frequent Words in Testing Dataset

Features

1. TF-IDF: A TF-IDF vectorizer was used with 1000 maximum features. The fundamental premise driving TF-IDF is that a term's significance is inversely correlated with its frequency across texts. A term's frequency in a document is revealed by TF, while its relative rarity within the collection of documents is revealed by IDF. Our final TF-IDF value can be obtained by multiplying these numbers collectively. The more relevant or vital a term is, the higher its TF-IDF score; as a term becomes less relevant, its TF-IDF score will decrease until it is zero.

$$tf\ idf\ (t, d, D) = tf\ (t, d)\ .\ idf\ (t, D)$$

2. Bag of Words (BoW): The BoW vectorizer used also had 1000 maximum features. A textual illustration of word recurrence in a document is called a "bag of words;" where attention to grammatical conventions or word order is ignored; and only word counts are kept track of. It is referred to as a "bag" of words because any details regarding the arrangement or structure of the words within the document are ignored. The model doesn't care where in the document recognized terms appear; it is only interested in whether they do.

3. Word2Vec: For more parity, a skipgram-type Word2Vec model of 1000-sized vectors and window size of 5 was trained. For these, the 1000 most frequent words were selected as feature representations. Word2vec, a two-layer neural network, uses word "vectorization" to parse text. A text corpus serves as its input, and its output is a collection of feature vectors, which stand in for the words in the input corpus. Word2vec converts text into a numerical form that deep neural networks can understand even if it is not a deep neural network. Beyond simply analyzing naturally occurring sentences, Word2vec has many other uses. It can also be used to identify patterns in genes, codes, likes, playlists, social media graphs, and other verbal or symbolic sequences.

Classification Models

1. K-Nearest Neighbors Classifier: K-nearest neighbors (KNN) is a supervised learning technique used for both regression and classification. By calculating the distance between the test data and all of the training points, KNN tries to predict the proper class for the test data. Then choose the K spots that are closest to the test data. The KNN method determines which classes of the "K" training data the test data will belong to, and the class with the highest probability is chosen. The

value in a regression situation is the average of the 'K' chosen training points. The KNN model used for this project had *'n_neighbors' =11*, which was obtained via *GridSearchCV*.

2. Multinomial Naïve Bayes Classifier: naive bayes is A probabilistic approach to creating data classification models which is often used as an alternative to decision tree forests and distance-based K-Means clustering. It is expressed as a set of algorithms which discusses probability as the probability that data falls inside a specific category. The classification of data that cannot be described quantitatively is possible with the multinomial model. The model used had a smoothing factor of 0, i.e., *"alpha" = 0.*

3. AdaBoost Classifier (of Linear Classifiers): AdaBoost, also known as Adaptive Boosting, is a machine learning method used in an ensemble setting. Decision trees with one level, or Decision trees with only one split, are the most popular algorithm used with AdaBoost. Another name for these trees is Decision Stumps. This algorithm creates a model while assigning each data piece an equal weight. Then, it gives points that were incorrectly categorized larger weights. The next model now gives more weight to all the points with higher weights. If no lower error is received, it will continue to train the models. For the project, the AdaBoost model used had 150 linear classifiers as estimator, and a learning rate of 0.1.

4. Decision Tree Classifier: By dividing the source set into subgroups based on an attribute value test, a decision tree can be "trained". It is known as recursive partitioning to repeat this operation on each derived subset. When the split no longer improves the predictions or when the subset at a node has the same value for the target variable, the recursion is finished. Decision tree classifier building is ideal for exploratory knowledge discovery because it doesn't require parameter configuration or domain understanding. High-dimensional data can be handled via decision trees. Decision tree classifiers are often accurate. A popular inductive method for learning classification information is decision tree induction. The model used had a maximum depth of 10.

Results

This section details the methods, models and training times obtained in the code.

Feature 1: TF-IDF

Lyrics with Stopwords		KNN	Naïve Bayes	AdaBoost	Decision Tree
	Train Accuracy	0.394	**0.405**	0.398	0.344
	Validation Accuracy	0.287	**0.384**	0.371	0.300
	Test Accuracy	0.271	**0.366**	0.363	0.300
	Training Time	1:36s	**279 ms**	2:39s	12.5s
Lyrics without Stopwords	Train Accuracy	0.404	**0.415**	0.40	0.333
	Validation Accuracy	0.266	**0.388**	0.383	0.296
	Test Accuracy	0.265	**0.368**	0.359	0.288
	Training Time	1:25s	**250 ms**	1:41s	7.87s

Table 3 Results with TF-IDF as Feature

Feature 2: BoW

Lyrics with Stopwords		KNN	Naïve Bayes	AdaBoost	Decision Tree
	Train Accuracy	0.394	0.409	**0.434**	0.354
	Validation Accuracy	0.280	0.354	**0.376**	0.305
	Test Accuracy	0.272	0.372	**0.387**	0.315
	Training Time	1:00s	**1.08s**	5:55s	9.88s
Lyrics without Stopwords	Train Accuracy	0.362	0.420	**0.427**	0.413
	Validation Accuracy	0.238	0.366	**0.377**	0.316
	Test Accuracy	0.237	0.365	**0.373**	0.290
	Training Time	53.9s	**254 ms**	2:53s	8.23s

Table 4 Results with BoW as Feature

Feature 3: Word2Vec

Lyrics with Stopwords		KNN	Naïve Bayes	AdaBoost	Decision Tree
	Train Accuracy	0.375	0.178	0.253	**0.442**
	Validation Accuracy	**0.251**	0.178	**0.251**	0.237
	Test Accuracy	**0.254**	0.177	**0.254**	0.246
	Training Time	25.3s	**1.5s**	9:00s	45.1s
Lyrics without Stopwords	Train Accuracy	0.373	0.178	0.246	**0.426**
	Validation Accuracy	0.246	0.182	**0.254**	0.240
	Test Accuracy	**0.260**	0.173	0.239	0.248
	Training Time	20.3s	**1.48s**	9:17s	46.4s
Notebook Runtime					53:17s

Table 5 Results with Word2Vec as Feature

Conclusion

Overall, the naive Bayes model outperformed the other models while requiring much less training time. The AdaBoost model, which has the drawback of having a lengthy training period, closely follows at second best. The TF-IDF representations of the lyrics scored somewhat higher than the BoW representations for the features. But for all models, the Word2Vec representations gave very subpar performance. The data also demonstrate that, despite the fact that the most common lyric words are stopwords, eliminating them barely affects the results.