Authorship attribution – Song Lyrics by Artist, Genre

Math 60629 – Machine Learning - Group Project

Baron, C. Malaguti, PFM. Wyman, E.

Contents

Introduction

What makes a Taylor Swift song a Taylor Swift song? What makes a Beatles song a Beatles song? Is it only a musical rhythm, an instrument, or can it be something else?

In the age of the internet many artists stand out with a particular visual identity and catchy melodies, the focus no longer seems to be on the lyrics of a song.

Nevertheless, the question this paper seeks to answer is: **Is it possible to predict the author or genre of a song by its lyrics?**

In order to answer this question a total of 18,000 unique artists across different time periods, and 9 unique genres were considered.

This project aims to employ various machine learning technologies, including supervised classification algorithms. An exploration of the data, and its structure and characteristics will proceed the design discussion. The experiment design based on intuitions and assumptions of song lyrics and algorithms will be outlined. Follows will be a description of the strategy of approach, including the selection, implementation and evaluation of several models. Finally, the results of the experiments, and their findings and implications on the project will be presented, with potential next steps and improvements identified from promising adjacent work in the field.

Data

This part of the report outlines the data explorations taken, the preprocessing steps and the challenges encountered through these activities.

The data set contains observations on artists, lyrics and genres collected from a website called lyrics.com. Lyrics.com is a community-based website whose operations are similar to a wiki page, with the aim of providing song lyrics, artists and other relevant data.

All the collected observations are divided into two data files. The first set contains 1,123,907 lyrics and artist information. The second set provides details on the genre of 211,032 songs. The gap between the two sets is due to inconsistency of the website in the recording of the genres per song.

Early in the project it was decided by the team to add some engineered covariates to see if they could help a model predict the genre or artist of a song. The new features are:

- Song runtime: the intuition behind this feature is that the length of a song can be very specific to a particular artist or genre. For example, a classical music concerto has an average length of 20-30 minutes, whereas a pop song is closer to 3 minutes.
- Song year: it is assumed that some genres or artists may have more present in some specific time periods.
- Song word count: the assumption is that some genres or artists characteristically sing more words per song than others. For instance, rap songs might have more words than jazz or pop songs.

After creating these variables came the preparation of the lyrics data. For the models to properly process the lyrics (as words), lyrics have to be transformed into tokens. In the project two types of tokenizers have been used. The first one, a count vectorizer, is used to create a bag of words, where each occurrence of a word is marked as +1. The other vectorizer used is the term frequency-inverse document frequency (TfIdf for short) vectorizer. Unlike the count vectorizer, this method allows for an equal comparison of texts of different sizes by considering the frequency of occurrence of a word relative to the length of the song (in words). The vectorized word representations were created and saved to different data sets, to be used as inputs for the models.

Another assumption relative to the lyrics is the application of ngrams. In the vectorizer algorithms ngrams define how many words are used for 1 token. By default, vectorizers use 1 ngram so that 1 word is equal to 1 token. However, in this project it was thought that some group of words might have more meaning than an individual word. For example, the model tries to see if “I love you” (ngram = 3) have more significance in the prediction than “I”, “love, “you” separately (ngram = 1).

During the data exploration phase, an important challenge was the abundance of duplicated songs. Indeed, despite having ~1.3 million lyrics at the beginning, only ~200,000 were unique in the end. A cause for this issue were duplicates, or near duplicates. After the duplicates were removed there was a significant loss of data.

After performing data preprocessing and cleaning, genres were merged to artist name in order to apply genre labels to the song lyrics data. The results were 151,275 labeled artist datum and 14,000 labeled genre datum.

Approach

The initial approach for the project was to gain insight into the differences between two seemingly similar tasks. Specifically, is predicting a song genre significantly different from predicting a song's artist, given English song lyrics. Through the course of this project, the goal was thus to determine differences in data representations, architecture and hyperparameters that specialized in these tasks, in order to gain insight into the inner workings of the classification models.

Due to the limitation of the genre-labeled data, as illustrated in the Data section, namely the constraint of ~14,000 labeled genre datum, the design of the experiments were modified. The reason for this was that this small amount of training data could introduce bias into the models, as this sample is likely not representative of the population.

Although the initial focus was to look at the differences between these tasks, a reasonable assumption can be made that they are similar in nature, as the tasks are both classification tasks, given text-based data, where the predictor variable is a categorical container for the data. The trade-off is the loss of granular exploration of the models, for the understanding of the robustness of the models.

The new approach was as follows. Run the modeling tasks using the abundant labeled data, in this case, song lyrics labeled with artist name. Through these initial experiments, the following are identified: the efficacy of supervised and unsupervised machine learning algorithms on the prediction task; hyperparameters configurations most beneficial to the model accuracy; and lastly, the ideal data format to feed these models for predictive purposes.

It has also been established that in performing further activities with this data, it is beneficial to obtain a fully labeled data set. This task became a real-world business problem therein, in populating the missing genre fields on lyrics.com.

Model selection

Why supervised models?

The intuition here is simple, being that the labels (artist name, and genre) are beneficial to the training and evaluation of the models. If the intuition was that these labels did not add value to the predictive task, unsupervised models would be more appropriate and the labels could be ignored.

Why was KNN used?

The first model tried was K nearest neighbor, or KNN for short. This model is rather simple due to the lack of parameters. It categorizes variables in groups based on their distance : the closest variables are grouped together.

This algorithm is simple and yet can be very powerful. Hyperparameter configurations are more easily understood, such as the ideal value for k, the number of neighbors to consider when classifying an instance of input, X.

The downsides of implementing a nearest neighbor model is that it requires storing the whole dataset, and the dataset is large. In addition, predictions over new data, of which there are many, are costly (slow), and these prediction operations will be performed many times through the experiments. It is also known that with high-dimensional data, as in this project, it might not be possible to see good accuracy as the model can select neighbors that are far away (although still closest).

K is the only hyperparameter needing to be tuned in a nearest neighbor model. It is the number of nearest neighbors used to identify the class of a new point. In the case of k = 1, the predicted class of the new point is simply the class of the closest 1 point. In the case of k = 2 or more, the decision is made using a majority vote based on the classes of the two or more nearest neighbors.

P is the distance function used to determine proximity between neighbors. With Euclidean distance issues may arise when data is highly dimensional which is the exact situation after encoding the data. Dimensionality concerns can be addressed by using either cosine or manhattan distance functions. Cosine distance is good for highly dimensional data but problematic as it doesn't factor magnitude. Manhattan distance also works well with high dimensionality and is good for scenarios where it is only possible to move between points in right angles. Through the experiments conducted, the Manhattan distance measure will be used.

The tradeoff for this algorithm is simplicity and understanding for storage cost, compute time and potentially performance.

Why neural networks?

Neural networks show great promise when inputs are high-dimensional and discrete, which is the case for the vectorized text as well as the song metadata included as covariates. Neural networks also perform well when the output is discrete, in our case, the classification problem has this

property. These models are also known to be robust and capable of learning complex patterns within the input data. This data is likely noisy, with many words being used in songs for reasons that aren't specific or unique to the artist or genre.

It is clear with the introduction of neural networks to the problem, that the modeling intuition is trading complexity, inference and difficulty of tuning for robustness and suitability to the data.

Combining models

One of the major weaknesses of the k-nearest neighbor algorithm is highly dimensional inputs, which happens to be a strength of the neural networks. An additional strength of neural networks is dealing with discrete outputs. The combination of these models could prove advantageous. Utilizing neural networks to reduce the dimensionality of the data, for input into a k-nearest neighbor model.

It should be mentioned that the implementation of recurrent neural network classifiers used was the MLPClassifier (multilayer perceptron classifier) from the scikit-learn neural network library.

Why a multilayer perceptron implementation of neural networks?

Multilayer perceptrons, commonly referred to as feed-forward neural networks, work well in classification scenarios where inputs are assigned labels. They are very flexible and can be used generally to learn a mapping from inputs to outputs.

Neural networks, multilayer perceptrons in this case, are highly configurable, therefore the hyperparameters selected within our model are as important as the structure itself. Following is an overview of the hyperparameters chosen in this problem context as well as justifications as to the values chosen. In all cases, performance will be measured using the accuracy measure of correct classification.

Hidden layers are the layers in a neural network, between the input and output layers, these hidden layers make up the "black-box" processing that is synonymous to modeling with these structures, although what happens in these layers cannot be inferred or observed, it is where the most significant processing takes place. The consensus on the number of these layers included in the models, as well as the number of nodes within said layers is not clear. Therefore, it is one of the hyperparameters to vary and select based on accuracy.

As with the number of hidden layers, the number of neurons in this layer will be varied to simulate wide and deep networks, again, selecting for accuracy.

A common hyperparameter used in neural network algorithms is non-linear activation functions for both the hidden and output layers of the neural network. Activation functions are used to better represent the data in a non-linear manner, for instance using softmax equations. This allows the network to accurately represent non-linearly separable data.

For the activation function for the hidden units, the "standard" ReLU function will be used, which also happens to be the default for scikit-learn's MLPClassifier. Taking advantage of the

observed improvement in results and faster training seen with this activation function due to its composition of two linear functions.

The activation function for the output layer is the softmax function, since the project's output type is categorical as it is predicting either the class of artist, or genre which the song lyrics belong to.

In similar neural network predictions over text data, it was observed that the Strength of the L2 regularization term, alpha, to be ideal at 0.3.

In the same context, an ideal learning rate was found, to be used as the default learning rate in the model, at a constant value 0.1.

Additional regularization methods can be helpful to address the problem of model overfitting. In the event of this trend in results, a regularization method, in the form of dropout, will be added in an attempt to remedy this issue.

Early stopping will be implemented, which allows the termination of the training of the network when the validation accuracy is not improving.

Batch sizes determine the number of samples to use before adjusting parameters, the use of a large number of samples effectively means large gradient steps. This value will be varied in the models, starting with an accepted default of 32, and increasing, in powers of 2.

When using the stochastic gradient descent for weight optimization, the number of iterations used will be the training epochs. An option for this value is to set it to a very high value, and employ early stopping to end the training period when validation accuracy fails to improve for more than two consecutive epochs. It is also common to use epochs equal to twice the number of inputs, in this case the number of inputs is equal to ~2000. Therefore, a reasonably high number of epochs should be sufficient at 6000.

From the model default, the tolerance for the optimization is decreases, allowing the model a longer time to converge,

To recap, most hyperparameters values of the multilayer perceptron (neural network instance) model will be fixed. These fixed values come from industry standards, recommendations, or algorithm defaults. The following hyperparameters will be varied where ideal values are less agreed upon.

Number of hidden layers, equal to 1, 2 or 5.

Number of nodes in the hidden layers, consistent for all hidden layers at either 10 or 100.

Batch size for the models, equal to 32, 64 or 128.

Model, hyperparameter and data evaluation

If the purpose of machine learning is to predict given new data, what is the baseline for improvement? In the context of these experiments, initially this baseline is random chance. This value is equivalent to picking the correct artist name, or the correct genre out of a hat, per se. Here are the established accuracy baselines for a random choice.

Artist name = 1 / 18280 = 5.47e-5 = 0.005%

Genre = 1 / 9 = 11.1%

The first intuition was to see if the use of supervised learning models (the use of labels) aided in the performance of the classification models, meaning increased their accuracy. A comparison of an unsupervised k-means model to a supervised k-nearest neighbor model was performed using the most basic data representation (bag of words only). As the labels assigned to a cluster by k-means are arbitrary it is not possible to compare them directly to the supervised model results. What was done instead was to use something called the Average Clustering Accuracy measure, this allows the calculation of an accuracy measure, based on the members of a cluster being together. With this test it was possible to compare performance against the supervised model which was found to outperform the unsupervised model, as expected.

The next steps were to identify hyperparameters and data representation for the simpler k-nearest neighbor model. As the data representations had less perceived variation than did the choice of neighbors (k), tuning the hyperparameter was done first. Again, using the basic bag of words input, values of k were evaluated using validation accuracy, and an ideal k for the artist name prediction task was found, of k = 1.

What this value suggests, in the context of this task, is that an artists' songs are most closely related to each other, in other words, they don't vary much. This goes against intuition as many artists use different words in their songs, and different artists use the same choice of words (think Christmas songs). Nevertheless, this is what the validation accuracy supports and this hyperparameter value will be fixed at k = 1.

The next task, using the 1-nearest neighbor model, was to attempt to identify the best data format for the prediction task. All combinations of the input covariates were fed to the 1-NN model whilst comparing validation accuracy. Recall, the covariates were either bag of words or TfIdf; the inclusion of all or none of the song metadata, in the form of song year, song runtime and song word count; and the inclusion or not of 3-words n-grams. The results of this experiment were that a simple bag of words representation of song lyrics along with song metadata performed best.

This supports the assumption that the term-frequency encoding of the lyrics removes information. In explanation, the repetition in songs can be significant to their style, and thus artist and or genre. Song metadata are seemingly informative as well, in terms of song year compared

to artist, as well as length and word count. Lastly, it could be that 3-word n-grams were an insufficient choice of length, or that this value is too varied on our set as a whole.

Now that the data representation has been chosen, it is possible to constrain the task of hyperparameter tuning on the neural network by fixing the data representation and comparing the performance of implementations with varying hyperparameters. Accuracy scores showed that a narrow (vs. wide), shallow (vs. deep) network architecture performed as good as or better than wider and/or deeper configurations, and that the smallest batch size performed just as well as or better than larger batch sizes depending on the shape of the network. The takeaway here is that the task was not a significantly complex one for the model to learn, or at least learn its best representation of.

At this stage in the experiments, the architecture and parameters for the prediction of genre are known. This having been performed on a dataset large enough to not add significant uncertainty to the results and conclusions.

Results

Captured in the table below is a subset of the model results used for model and hyperparameter selection, as well as model evaluation.

Note: rows in bold indicate best result of motivation (task).

Note: the percentage of variables used in each partition is in italics

Motivation	Model	Covariates (X)	Hyper-parameters	Pred (y)	Training set performance Accuracy (size)	Validation/ test set performance Accuracy (size)
Baseline (unsupervised)	K-means	BOW	N_clusters = 9	Genre	0.237426 (*80%*)	0.029586 (*20%*)
Baseline (supervised)	**K-means**	**BOW**	**K=100**	**Genre**	**0.115385** **(*80%*)**	**0.076923** **(*20%*)**
Tuning	**KNN**	**BOW**	**k =1** **p = 1**	**Artist**	**0.99686** **(*4%*)**	**0.018506** **(*1%*)**
Tuning	KNN	BOW	k = 64 p = 1	Artist	0.049091 (*4%*)	0.009914 (*1%*)
Tuning	KNN	BOW	k = 128 p = 1	Artist	0.035041 (*4%*)	0.006609 (*1%*)
Tuning	KNN	BOW	k = 256 p = 1	Artist	0.024132 (*4%*)	0.006609 (*1%*)
Covariate, model consideration		**BOW, metadata**	**k =1** **p = 1**	**Artist**	**0.999207** **(*20%*)**	**0.046668** **(*5%*)**
Covariate consideration		BOW, metadata ngram=3	k =1 p = 1	Artist	0.999022 (*20%*)	0.040365 (*5%*)

Covariate consideration		TFIDF, metadata	k =1 p = 1	Artist	0.999207 (*20%*)	0.02036 (*5%*)
Covariate consideration	KNN	TFIDF, metadata ngram=3	k =1 p = 1	Artist	0.999207 (*20%*)	0.02036 (*5%*)

Hyper-parameter tuning;
Model consideration
	RNN	Bag of words; meta data	Epochs = 6000
			#Hidden = 1
			Batch = 32
			#Neurons = 10
				Artist	0.003933
					(*20%*)
						0.005288
						(*5%*)

Hyper-parameter tuning;
Model consideration
	RNN	Bag of words; meta data	Epochs = 6000
			#Hidden = 2
			Batch = 128
			#Neurons = 100
				Artist	0.003933
					(*20%*)
						0.005288
						(*5%*)

Final model evaluation	**KNN**	Bag of words; meta data, artist name	**k=1** **p=1**	**Genre**	**0.795627216** **(*80%*)**	**0.211343687** **(*20%*)**

See appendix A for definitions used in the table.

Summary

Three main assumptions were made about the similarity of the prediction problems, recall these as the prediction of song genre given song lyrics, and the prediction of artist name given song lyrics. This section is concerned with the verification of these assumptions, numbered below.

1. Is the hyperparameter (k=1) identified for predicting artists, the best choice when the problem changes to predicting genre?

No, it is not, as seen by the supplementary experiments performed, with results shown below.

Model	Covariates (X)	Hyper-parameters	Pred (y)	Training set performance Accuracy (*size*)	Validation/ test set performance Accuracy (*size*)
KNN (best)	**Bag of words; meta data, artist name**	**k=1 p=1**	**Genre**	**0.79563 (*80%*)**	**0.21134 (*20%*)**
KNN	Bag of words; meta data, artist name	k = 10 p = 1	Genre	0.45456 (*80%*)	0.319518 (*20%*)
KNN	Bag of words; meta data, artist name	k = 32 p = 1	Genre	0.41586 (*80%*)	0.36964 (*20%*)
KNN	Bag of words; meta data, artist name	k = 64 p = 1	Genre	0.39718 (*80%*)	0.37108 (*20%*)
KNN	**Bag of words; meta data, artist name**	**k = 100 p = 1**	**Genre**	**0.39019 (*80%*)**	**0.37398 (*20%*)**
KNN	Bag of words; meta data, artist name	k = 500 p = 1	Genre	0.37532 (*80%*)	0.36241 (*20%*)
KNN	Bag of words; meta data, artist name	k = 1000 p = 1	Genre	0.36258 (*80%*)	0.3441 (*20%*)

2. Is the best model for artist name prediction (KNN, with k=1) the best model for genre prediction?

   Yes, when the improved accuracy of K=100 for the KNN classifier is factored in, it beats out the NN model.

Model	Covariates (X)	Hyper-parameters	Pred (y)	Training set performance Accuracy (*size*)	Validation/ test set performance Accuracy (*size*)
KNN	Bag of words; meta data, artist name	k = 100 p = 1	Genre	0.39019 (*80%*)	0.37398 (*20%*)

RNN	Bag of words; meta data	Epochs = 6000
		#Hidden = 1
		Batch = 32
		#Neurons = 10
			Genre	0.261945 (*80%*)	0.256921 (*20%*)

3. Is the data representation for artist name prediction (bag of words, and song metadata) the best data representation for genre prediction?

   The validation of this assumption could be found, for example by fixing k=100, and analyzing the results of different formats of X. This task was not performed.

Conclusion

In this project, the goal was to explore the task of authorship attribution for artists and genre based on song lyrics. It can be concluded through the experiments conducted that this classification task is appropriate for supervised machine learning algorithms. The KNN model that was trained demonstrates that songs with the same artist typically tend to have similar lyrics, whereas the lyrics used in the same genre differ more.

Assumptions were made about the similarity of tasks within the context of artist attribution. The results seen inform that these models are much more complex than a naive intuition might suggest. Through this project it was found that seemingly similar tasks are represented and thus learned by models in many different ways, and that the black box characteristic of these algorithms holds true for a reason.

This leaves a fantastic opportunity for more research to be done in this area. More flexible sequential models such as recurrent neural networks and more notably Transformers (Vaswani et al., 2017) can potentially see improvements in terms of accuracy of authorship attribution. The transformer architecture has been shown to perform well on classification tasks (Sun et al., 2020). The goal of the transformer model is to accurately predict the task by identifying the most key features in its input (Attention). The main components of the self-attention mechanism which are utilized in the Transformer architecture can be summarized as (Amini ., 2022):

1. Encoding of position information
2. Extraction of query, key, and values for search
3. Computation of attention weights
4. Extraction of features

By adding this mechanism, the model can focus on the most key features in the dataset. In a way, there becomes a better understanding of the context of the inputs.

Transformers have seen tremendous success in recent years with the advancements of BERT by Devlin et al. (2019), GPT-3 (Brown et al.), and more recently ChatGPT. It will be exciting to see what the future holds for authorship attribution in music and in more general contexts. Potentially one day, everyone will be able to write lyrics with the political style of Bob Dylan or with the pop “catchiness” for the radio.

Sources

Alexander Amini. (2022, March 18). *MIT 6.S191: Recurrent Neural Networks and Transformers* [Video]. YouTube. https://www.youtube.com/watch?v=QvkQ1B3FBqA

Brownlee. (2022, August 12). *When to Use MLP, CNN, and RNN Neural Networks*. Machine Learning Mastery. https://machinelearningmastery.com/when-to-use-mlp-cnn-and-rnn-neural-networks/

Charlin, M.-M. (2020a, September 24). *Week 5 -- Capsule 4 -- Neural Network Hyperparameters* [Video]. YouTube. https://www.youtube.com/watch?v=5axp1O299qM

Charlin, M.-M. (2020b, September 24). *Week 5 -- Capsule 5 -- Neural Network Takeaways* [Video]. YouTube. https://www.youtube.com/watch?v=Nqs-C7wBVQo

Chowdhury, K. (2022, November 7). *10 Hyperparameters to keep an eye on for your LSTM model — and other tips*. Medium. https://medium.com/geekculture/10-hyperparameters-to-keep-an-eye-on-for-your-lstm-model-and-other-tips-f0ff5b63fcd4

How to test accuracy of an unsupervised clustering model output? [Online forum post]. (2017, March 9). Data Science Stackexchange. https://datascience.stackexchange.com/questions/17461/how-to-test-accuracy-of-an-unsupervised-clustering-model-output

Kondyurin, I. (2023). *Explainability of Transformers for Authorship Attribution* (D. Paperno & Y. Du, Eds.) [MA thesis]. Utrecht University.

sklearn.neural_network.MLPClassifier. (n.d.). Scikit-learn. https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPClassifier.html

Sun, C. (2019, May 14). *How to Fine-Tune BERT for Text Classification?* arXiv.org. https://arxiv.org/abs/1905.05583

Vaswani, A. (2017). *Attention Is All You Need* (I. Polosukhin, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. Gomez, & L. Kaiser, Eds.; 31st Conference on Neural Information Processing Systems). https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf

Wyman, E. (2022) Machine Learning for Large-Scale Data Analysis and Decision Making. MATH 60629A Homework Assignment

Appendix A – Results definitions

Definitions:

- Models
  - KNN – K-nearest neighbor model, K given by hyperparameter declaration
  - RNN – Recurrent neural network
- Covariates (X)
  - BOW: bag of words
  - TFIDF: frequency–inverse document frequency
  - metadata:
    - song_runtime: song runtime (in seconds)
    - song_year: song release date (year only)
    - song_word_count: number of words in song
- Hyperparameters
  - N_clusters: number of clusters used in K-means classifier
  - K: number of neighbors used in K-nearest neighbor algorithm
  - p: distance function used in K-nearest neighbor algorithm
    - p=1: Manhattan distance
  - #Hidden: number of hidden layers in neural network
  - Batch: batch size
  - #Neurons: number of neurons in each hidden layer
- Size: Percentage of data used in iteration