Authorship Attribution
Natural Language Understanding Report

Harvey South **Christopher Norman**

Abstract

Authorship attribution is a multi-class classification problem with the aim to find the author (from a preset list of authors) for a given piece of text. It is applied to a range of tasks such as bibliometrics and plagiarism detection. In this paper we implement and compare LSTM, GRU and Siamese implementations for authorship attribution on a data set with preexisting accuracy benchmarks. These neural network architectures were then applied to attribute a musical lyric document to an artist with varied success. There was evidence to suggest that the trained models could be used to create a music recommendation system based on only lyrics. In future our music recommendation model could be adapted to improve current systems.

1 Problem and Motivation

A challenge in authorship attribution is effectively capturing features of a person's writing (*stylometric features*) in a piece of text. In (Stamatatos, 2009) they divide *stylometric features* into six types: character, lexical, syntactic, structural, semantic and application-specific features. To obtain all the information from an author's writing style these features must be transformed into a vector without much loss of information. Furthermore, in complex applications, there can be thousands of different authors to classify which makes authorship attribution a difficult task. In the first half of this paper we will implement and compare GRU, LSTM and Siamese networks to existing benchmarks.

Authorship Attribution (AA) is a task that can be applied to a range of applications: plagiarism detection (Stamatatos and Koppel, 2011), verification of claimed authors for historical texts (Berton et al., 2016), or even for forensics (Ainsworth and Juola, 2018). Therefore an interesting question is, can AA be applied to the lyrics of song artists?

A large proportion of songs contain lyrics and are an incredibly important device within music for conveying emotion and expression. MIDiA Research, a research and analysis company, report there were around 523.9 million music subscribers in the first quarter of 2021 (Mulligan, 2022). These streaming services gather a huge amount of data on music and users which is used to drive new growth. Spotify has the highest market share of 31% (Mulligan, 2022). A main feature of Spotify that differentiates it from other music sharing services is the excellent music recommendation, used in '*Daily Mixes*', '*Song Radios*' and '*Autoplay*'. In the second half of this project we will adapt the neural network architectures to classify song lyrics to their artist and produce a vector representation for each song for use in a song recommendation system.

2 Authorship Attribution on IMBD62 Dataset

In this section we discuss and evaluate LSTM, GRU and Siamese architectures for attributing authors to a review from the IMBD62 data set (Seroussi et al., 2014). We then compare them to the baselines stated in (Fabien et al., 2020).

The models created for the task largely follow the works of (Qian et al., 2017) where the model architecture finds success on datasets for news articles and for novels.

2.1 Data Preprocessing

The IMDB62 dataset contains 62 unique authors each with 1000 movie reviews found on IMDb. Initially the reviews were a string of tokens with a space between them. We create a list of these tokens by splitting this string by the 'space' delimiter.

2.1.1 GloVe Vectors

We decided to use cased GloVe vectors for the word embedding inputs in the model. The GloVe embedding file is imported and indexed, we then convert each reviews list of tokens into a list of

indexes corresponding to the vector for each token in the GloVe file. This allows us to retrieve the vectors faster, reducing computation time. The IMDb user IDs are also mapped to indexes 0-61 to make classification easier. If a token does not exist in the GloVe file we replace it with an '<unk>' token which we map to a zero vector. We also initialise a '<pad>' token to zero to be used for padding short inputs.

2.2 LSTM Implementation

The implementation of the LSTM is as follows: text is preprocessed into identification tokens based on the vocabulary of the text where each unique token is given an identifier with an assigned embedding that is learned by the model. The embeddings created by the embedding layer are fed into a single layer LSTM. Average pooling is applied to every output of the LSTM, and then this pooled output is reduced or expanded to the target size which is the number of authors in the dataset. After producing the scores for each author with the linear layer, we apply log softmax normalization to the output, giving a probability prediction from the model for each author in the dataset. The model is trained to minimize the negative log likelihood between the author of the text and the model probability predictions. We use the Adam optimizer to update the weights of the model given the loss, early attempts at using stochastic gradient descent were deemed to be too slow for training. A diagram can be followed of the architecture in figure A.1.

2.3 GRU Implementation

The implementation of the GRU follows the same architecture as the implementation of the LSTM model. Both the GRU and LSTM are variants of the recurrent Neural Network, and are interchanged.

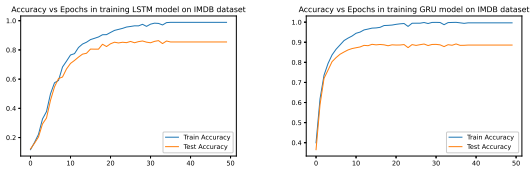
2.4 Siamese Implementation

The Siamese implementation for the task is trained on the model's ability to produce similar or dissimilar outputs based on two input text documents. This is achieved through running each of the two documents through the same GRU model (the model having the same model weights both times), with the architecture as described in the GRU implementation, separately. After the model has produced an output for each document, the loss that the model is trained to minimize is calculated to be the sum of the negative loss likelihood for the prediction of each output, added to the absolute difference between optimal cosine similarity and actually produced cosine similarity between the two outputs. Cosine similarity is bounded between zero and one, and so the loss generated by the similarity calculation is scaled by 5 in order to contribute more to the overall loss on a model prediction. The model is able to produce a prediction on singular documents by using the GRU trained in the similarity based loss calculation.

2.5 IMBD62 Evaluation

Model	Accuracy
LSTM	85.45%
GRU	88.60%
Siamese GRU	86.81%

Table 1: Testing accuracy for IMDB review author classification



LSTM GRU

Figure 1: Train and test classification accuracy for IMDB dataset vs epoch.

Trained on the training set of data, and evaluated on both the training and testing datasets every epoch Figure 1 displays the accuracy over each epoch. The final accuracy achieved after training on the test datasets are displayed in Table 1. Figure 2 shows the loss over 25 optimizer steps, as single document evaluation was not performed during training

These experiments were run to test the architecture on a dataset with a known learn-able relationship.

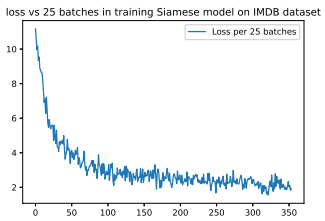


Figure 2: Siamese model loss per 25 optimizer steps.

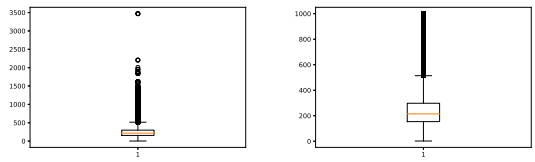


Figure 3: Original distribution of number of tokens in each lyric

2.6 Discussion of results

The results of all models on the IMDB datasets achieve an accuracy above the baselines for method comparison presented in (Fabien et al., 2020). The greatest baseline, using TF-IDF methods achieves an accuracy of 82.1% on IMDB authorship attribution of 50 authors, an easier task than the increased 62 authors where the produced LSTM, GRU, and Siamese trained GRU models achieve an accuracy above 85%. These results show evidence to suggest that the models have the ability to learn a relationship between text documents and their written authors. The Siamese model did not show improvement in training the GRU for single document classification.

3 Authorship Attribution for Lyrics

In this section we apply the earlier models to lyric author attribution and then look at the effectiveness of a music recommendation system using lyrics.

3.1 Preprocessing Lyrics

We use a data set (from Kaggle) that contains artists and their lyrics found on Genius (a song lyric database) from unique Spotify track ID's (Edenbd, 2020).

The lyrics found in this data set were already in a use-able format for this project and minimal processing is required. We first split the lyric into a list of tokens using a general tokenizer that retains capitalisation and punctuation. We maintain punctuation as tokens and capitalised letters to try and learn the general structure of each song and not overfit to the contents.

Figure 3 shows the distribution of number of tokens. As we can see this varies a lot and has a wide range with a number of outliers. To improve performance the number of tokens was truncated to 1000 whilst maintaining enough information from each of the lyrics as possible.

In total there are unique 14691 artists in the dataset. We decided to reduce this number to 1000 artists, if this step was not taken many classes would have only one or two samples which is not enough data to train on. Hence, we ordered the artists by number of songs and took the artists with the most songs. This resulted in a range of 20-880 samples per artist leading to a large data imbalance. This would skew the training and accuracy metric so we re-sampled each artists songs to 250 each using both over-sampling and under-sampling. In the end we obtained a dataset containing 250000 samples which we split into train:validation:test as 70%:15%:15% respectively. This data was then processed as described in Section 2.1.

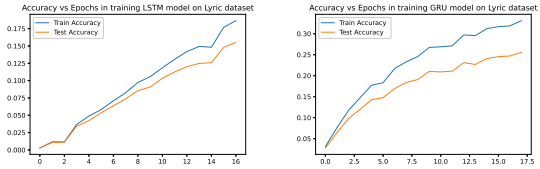
3.2 Lyric-Author Classification

We first train an LSTM model and a GRU model with the lyric data to compare accuracy for this dataset. Figure 4 shows plots of accuracy over the number of training epochs for both LSTM and GRU. We can see that even after a large amount of training both the test accuracy and train accuracy are still relatively low but GRU has roughly 10% better accuracy as shown in the Table 2. This is likely due to the GRU having fewer model parameters and hence is less likely to overfit the model to any noise in the training data due to having a small sample size for each artist (Burke and Ignizio, 1997).

Model	Accuracy
LSTM	15.17%
GRU	25.65%

Table 2: Testing accuracy for lyric classification with 100 artists

We also compare the classification accuracy for lyrics with number of artists $n \in \{8, 64, 1000\}$ using an LSTM in Figure 5. As it was much faster to train 8 and 64 classes we use more epochs. There are a few important features in this graph, the first being that up to 17 epochs $n = 1000$ has a better testing accuracy than $n = 64$ and secondly that the test accuracy does not exceed 40% even for a small number of classes and a long training time. The former could be explained by the choice of the subset of artists, the artists chosen were not from significantly different genres (Elvis Presley, Frank Sinatra, Johnny Cash, Ella Fitzgerald, etc.) so their



LSTM GRU

Figure 4: Train and test classification accuracy for lyrics and artists over the number of epochs.

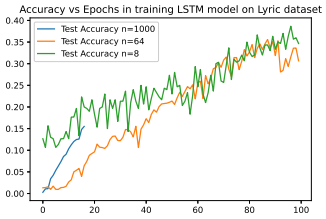


Figure 5: Test and train accuracy for different number of artists

lyrics may contain a large amount of overlapping subject matter making it hard for the model to learn to differentiate between artists. The latter seems to show evidence of underfitting and hence, the lyric data may not be suitable for this task. We also ignore the fact that not all artists write their own songs and many songs may be written by the same producer which could skew these results.

3.3 Music Recommendation

While the test classification accuracy reached a maximum of around 25%, we wanted to see if the trained model could be used for something other than authorship attribution. The aim for a music recommendation system is to assign a vector representation for each song and cluster or use k-nearest neighbours to find similar songs. For our model we train the classification model as before and obtain a vector representation for each song from the final hidden layer pooled output. We use the GRU architecture as it produced the highest classification accuracy.

3.3.1 Recommendation Evaluation

For this project it was not feasible to manually evaluate how effective the recommendations are. Instead we split the evaluation into two tasks and looked for evidence to suggest that the produced vector representations are suitable:

1. Artists the model was trained on - k-means clustering on a subset of artists (3 or more artists). We choose 2 theoretically similar artists and 1 different and test the hypothesis that the 2 similar artists are clustered together without the different artist when we use a value of $k = 2$ clusters.

2. Unseen artists - Run the model for an unseen artists songs and retrieve each vector representation. Take the mean vector value and use k-nearest neighbours return the k closest songs in the testing set (from artists used in training). We then check manually to see if the artists are in any way similar, e.g. genre, language, location and lyric subject matter.

```
Artist 1, The Rolling Stones :
[1 1 1 1 1 0 1 0 1 1 1 1 1 1 0 1 1 0 1 1 0 1 1 1 1 1 1 0 1
1 0 0 0 1 1 1 1]
Mean Value: 0.75

Artist 2, The Beatles :
[1 1 1 1 1 1 1 0 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1]
Mean Value: 0.9629629629629629

Artist 3, Maroon 5 :
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 1 0 0 0 0]
Mean Value: 0.025

SIMILARITY:
ARTIST 1 and ARTIST 2: 0.7870370370370371
ARTIST 1 and ARTIST 3: 0.275
ARTIST 2 and ARTIST 3: 0.06203703703703711
```

Figure 6: Example output for clustering vector representations of songs for 3 artists

One problem with this evaluation is how subjective it is and also that we make an assumption that an artist's lyrics usually have the same general subject matter and genre (whilst this is usually true it may not be for all cases). Figure 6 shows an output of clustering 3 pre-trained artists unseen music into 2 clusters (the vectors being the cluster labels for each song). The output shows that The Rolling Stones are mostly clustered with The Beatles and Maroon 5's lyrics are mostly clustered together. We take the mean class value and find the absolute difference between the other classes to find a 'similarity'. The output in this case is as expected. This process was repeated a few times for other artists, further manual testing is required to check if the model works well. In general, with unseen artists the k-nn results seemed quite random by inspection with a few songs in the top 30 closest being close in genre.

3.3.2 Music Recommendation Conclusions

The model showed evidence of underfitting the data a lot (large discrepancy between train and test accu-

racy and a low train accuracy). This is likely due to the very small number of samples per class and it is also possible that lyrics do not generalise well for authorship identification. The data is quite noisy and contains a large number of unknown tokens. As we have so many classes (a large number of model parameters) and so few samples this means that the model will be inherently biased. Other models could be created in the same way but classifying genre instead of author to reduce the class size and increase sample size. As a number of words were not in the *GloVe* vector dictionary we could use *fastText* for out-of-vocabulary word vector representations to improve the model.

A clear downside to this method for music recommendation is that not all songs have lyrics, hence, it would make sense to combine our model with some 'User Modelling' which updates and improves recommendations over time as a user either interacts positively or negatively to new suggested song (Mehrotra, 2021). It is also likely that this model would not scale well to such a large application servicing millions of users and artists.

However, there was some evidence to suggest that the model could provide a reasonable vector representation for a lyric especially if the model was trained on the artists work. For unseen artists (artists we did not train on) the model did not show signs of working well. The effectiveness of a song recommendation system must be manually evaluated as is hard to evaluate it objectively. Unfortunately, we did not have the time or resources to complete a comprehensive manual evaluation in this project. In the future we would also like to further tune the hyper parameters such as the learning rate and the total number of epochs.

References

Janet Ainsworth and Patrick Juola. 2018. Who wrote this: Modern forensic authorship analysis as a model for valid forensic science. *Wash. UL Rev.*, 96:1159.

Gary Berton, Smiljana Petrovic, Lubomir Ivanov, and Robert Schiaffino. 2016. *Examining the Thomas Paine Corpus: Automated Computer Authorship Attribution Methodology Applied to Thomas Paine's Writings*, pages 31–47. Palgrave Macmillan US, New York.

Laura Burke and James P Ignizio. 1997. A practical overview of neural networks. *Journal of Intelligent Manufacturing*, 8(3):157–165.

Edenbd. 2020. 150k lyrics labeled with spotify valence (version 1). https://www.kaggle.com/edenbd/150k-lyrics-labeled-with-spotify-valence. [Online; accessed 14-March-2022].

Maël Fabien, Esau Villatoro-Tello, Petr Motlicek, and Shantipriya Parida. 2020. BertAA : BERT fine-tuning for authorship attribution. In *Proceedings of the 17th International Conference on Natural Language Processing (ICON)*, pages 127–137, Indian Institute of Technology Patna, Patna, India. NLP Association of India (NLPAI).

Rishabh Mehrotra. 2021. *Algorithmic Balancing of Familiarity, Similarity, and Discovery in Music Recommendations*, page 3996–4005. Association for Computing Machinery, New York, NY, USA.

Mark Mulligan. 2022. Music subscriber market shares q2 2021. [Online; accessed 15-May-2022].

Chen Qian, Ting He, and Ren Zhang. 2017. Deep learning based authorship identification. In *CS224N Reports*.

Yanir Seroussi, Ingrid Zukerman, and Fabian Bohnert. 2014. Authorship Attribution with Topic Models. *Computational Linguistics*, 40(2):269–310.

Efstathios Stamatatos. 2009. A survey of modern authorship attribution methods. *Journal of the American Society for Information Science and Technology*, 60(3):538–556.

Efstathios Stamatatos and Moshe Koppel. 2011. Plagiarism and authorship analysis: introduction to the special issue. *Language Resources and Evaluation*, 45(1):1–4.

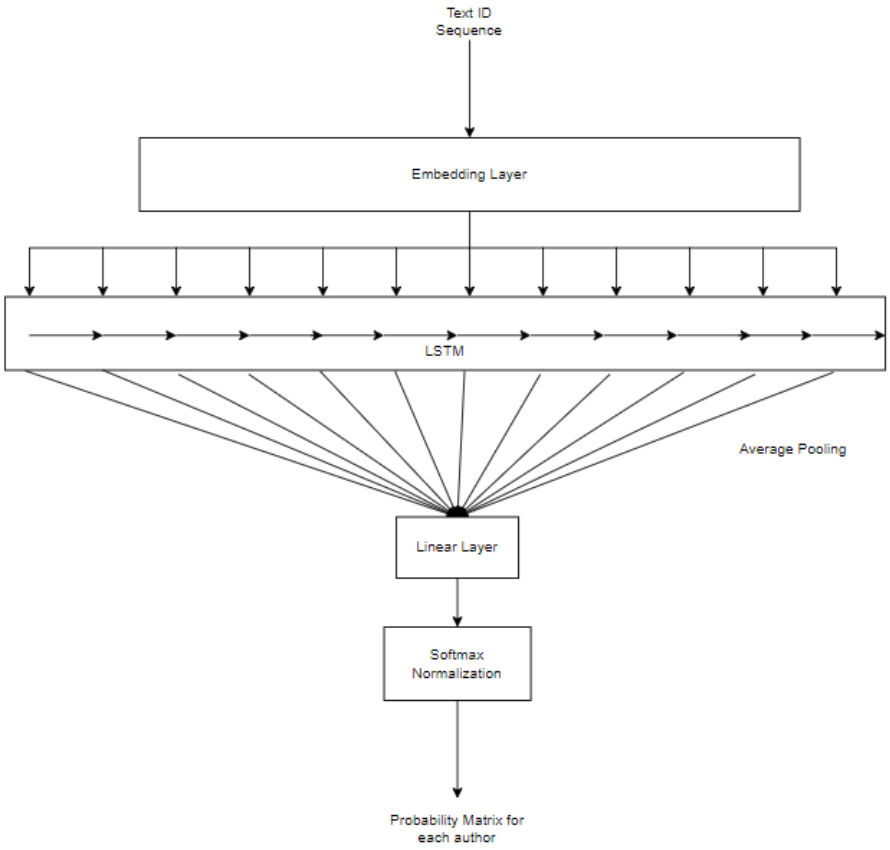


Figure A.1: Diagram for LSTM model architecture.

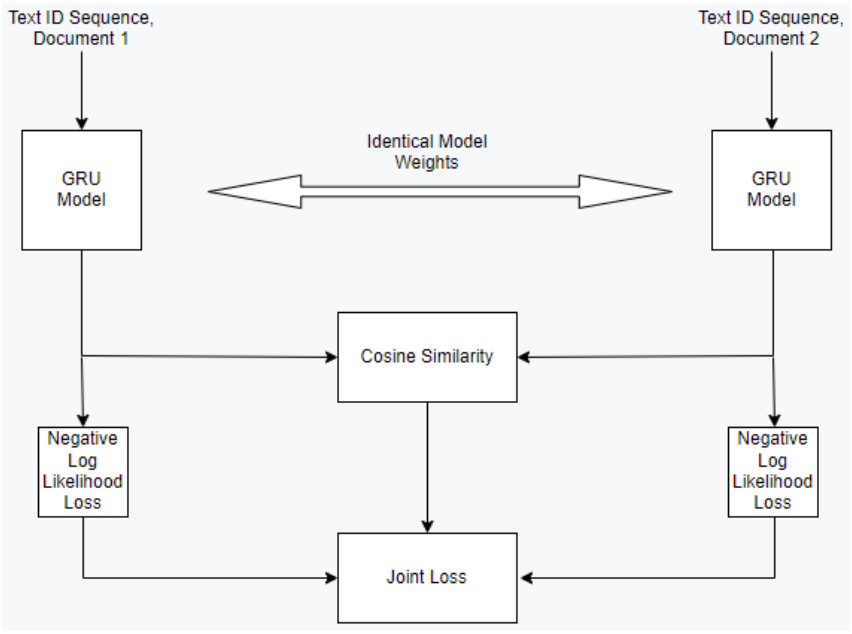


Figure A.2: Diagram for Siamese loss calculation.