COMP90051 Assignment 2

J. Cumming 995682 and **C. Holmes 899251** and **O. Bestel de Lezongard 914956**

1. Introduction

Authorship attribution (AA) is a sub-discipline of the wider field of authorship analysis, which focuses on analysis of data with known or unknown authors of interest and whose applications manifest broadly from digital forensics to intelligence. Specifically, the objective of authorship attribution is to disseminate documents of text written by distinct authors Stamatatos (2009).

Our challenge was a multi-label instance within this framework - given a training corpus of pre-processed research literature article titles, abstracts and other metadata, we were tasked to predict which of a set of 100 known "prolific" authors contributed. This report outlines our thought process, experimentation, and the final best-performing Fully-Connected Neural Network (FC-NN) outlined in 2.4. For the remainder of this report `Train[attr]` and `Test[attr]` refer to attributes of their corresponding dataset.

2. Methods

2.1 Data Pre-Processing

Below is a summary of the key pre-processing steps:

1. `Train[Authors]` was split into two features, `Train[Coauthors]` and `Train[Prolific]`
2. `Train[Prolific]` data was one-hot encoded into length-100 vectors, and assigned as the output labels
3. `Train[Coauthors]` and `Test[coauthors]` data were one-hot encoded as well
4. `Train[Venue]` and `Test[Venue]` missing values were replaced with a new -1 venue label (see 3.1)
5. A train-validation split was applied to `Train` to utilise data with `Venue` $= 18$ for validation.

An initial analysis of the data was done to inform how to deal with any missing data. Little's MCAR test was used to determine the nature of the missing data.

While word n-grams were considered as a potential feature, ultimately we chose to avoid techniques specific to Natural Language Processing (NLP), and instead opted for the (lower-dimensional) term-frequency inverse-document-frequency scores (TF-IDF). This model was tested by application to `title` and `abstract` features, and then applying k-means clustering on both (independently), assigning a new categorical feature to distill the textual data. The number of clusters was optimised independently for the abstract and title vectors using the elbow method.

2.2 Categorical Embedding

While `Train[Coauthors]` can be readily one-hot encoded for compatibility with machine learning models, there are over 20,000 coauthors identified; introducing feature vectors of this length significantly increases feature sparsity and hurts computational feasibility by enlarging the working model, which can lead to model underfitting.

To transform the coauthor data to a lower-dimensional feature space, two methods were considered. First, an autoencoder model (a simple feed-forward neural network with one hidden layer), and a categorical embedding model (a simple feed-forward neural network with two hidden layers). In both instances, the models were trained with batching and with 20%*dropout* on a train-validation split with binary cross-entropy loss, with hyperparameter tuning on the size of the final (embedding) hidden layer. Ultimately, little success was found with the autoencoding approach across a variety of loss functions, as the high sparsity of the one-hot data encouraged zero-predictions across all coauthors. Alternatively, the categorical embedding model suffered no issues, and moreover managed to attain a macro-averaged F_1 score of over $\approx 62\%$ in predicting `Train[author]` on held-out data.

A smaller one-layer embedding model of size 5 was learnt with the same model setup for `Venue`; this was done to generate a higher-dimensional representation (rather than dimensionality reduction) that could learn a non-linear relationship with associated prolific authors.

Thus, the categorical embedding model identified above was preferred.

2.3 Baseline Models and Other Models

Two baseline models were chosen - a Gaussian Naïve Bayes (GNB), and a Dummy Classifier predicting no prolific authors - the most frequent option.

A series of other models were tested. Namely, two ensemble learners, recurrent neural networks (RNNs), convolutional neural networks (CNNs) and a random forest (RF) model. The CNN and RNN frameworks were implemented in an attempt to learn sequential linguistic features in the `Author` and `Title` fields. The first ensemble learner used a neural network (NN) meta-classifier with inputs from a RF classifier and a categorical embedding classifier. The structure of these models will not be discussed further as they were not chosen due to poor performance. A stand alone RF model was trained with one-hot encoded `coauthor`, numeric `year`, and categorical `venue`. The optimal depth was trialled, and it was run with and without the k-means clustered `title` and `abstract` values.

2.4 Feed-Forward Neural Network with Embeddings

A fully-connected feed-forward neural network (FC-NN) was considered as a rudimentary model, where all 5 metadata would have dedicated sub-models feeding to a dense FC-NN with 1 layer, using a sigmoid activation function and binary cross-entropy to learn prolific authorship.

It was found that the inclusion of high-dimensional `title` and `abstract` features encouraged the FC-NN to default to predicting no prolific authors universally; this was attributed to the low volume of data available, and the demand for significant number of parameters to model associations with these data; thus, a model utilising only {`Coauthor, Venue, Year`} was designed.

The embedding models identified in 2.2 were used as sub-models for `Coauthor` and `Venue` respectively; pre-training for these weights (rather than re-learning these weights during training) was considered, but it was found to offer no greater performance at higher training time.

3. Results

3.1 Preprocessing and Hyperparameter Tuning

In the training data set, $1,578$ (6.1%) instances had a missing venue. No missing years were detected. Little's MCAR test determined that the data was not missing completely at random ($p < 0.001$), and so the decision was made to encode all missing venues as '-1'.

The TF-IDF vectors were of length 3,482 for the title model, and of length 4849 for the abstracts. The elbow method was chosen to decide the ideal k, as 100 for the title TF-IDF k-means clusting, and similarly 200 for the abstract clustering.

The optimal depth for the random forest was found to be 1866. It performed worse with the k-means clustered title and abstract values, and hence the final model used `venue`, `coauthor`, and `year` information.

3.2 Model Results

The results of the Kaggle submissions on the full test data set are summarised in the table below.

Model	Kaggle F1 Score
Categorical Embedding	0.50083
Random Forest	0.45541
Ensemble (NN Meta-Classifier)	0.45708
GNB Baseline	0.20250
Dummy Classifier (Most Frequent Strategy)	0.20500

4. Discussion

Although training a TF-IDF vectorisation of the titles and abstracts should highlight key features and unique words that might enable the identification of a specific author/authors, ultimately the full vectorisation had far too great of a dimensionality to be useful in and of itself. When trying to 'cluster' similar titles, it is likely that the variation is too great, thus the random forest model performed worse when trying to use the cluster categorical random variables.Similarly, the more complex RNN and CNN models failed to predict any true positives. This suggests that the clusters added meaningless noise to the data. It is likely that the TF-IDF vectors would have benefited greatly from dimensionality reduction. Unlike CNN and RNN embeddings, TF-IDF vectorisation ignores word order, a key linguistic identifier.

The FC-NN model outperformed the RF model (our second-best performing model) by $\approx 4\%$. We attribute this marginal gap in performance to the vastly higher dimensionality of the feature space used by the RF; even though RFs are highly effective at feature selection and do not suffer from the curse of dimensionality as severely, it is suspected the more informative representation of `coauthor` and `venue` used in the FC-NN gave it a performance edge.

4.1 Final Approach Motivation Evaluation Pros and Cons

Training with regularisation indicated somewhat minor performance gaps between training error and validation error ($< 1\%$), but large performance drops when applied to test data ($> 20\%$). This suggests that even with regularisation, there is significant distributional differences between the train and contest data, and the model is not sufficiently general. This may be addressable with additional data (i.e. `title` and `abstract` feature engineering / learning, such as an RNN sub-model or NLP features); use of early-stopping criteria (which, by terminating model training early, prevents model overfitting and diminishing returns on computation); and/or harsher dropout and normalisation and other regularisation techniques.

However, the model was quite robust when comparing the private and public competition scores (a drop of only about 1.3%) - evidence that our methodology did not overfit to the test data. Other sources of improvement likely include improving the loss of information in the embedding sub-model for `coauthor` (see 2.2). As highlighted above, the strengths of the FC-NN lie in its simplicity. The efficient dimensionality reduction on `coauthor` allows a relatively low-dimension, highly informative feature space.

5. Conclusion

Overall, we had a decent performance but there was clear room for improvement with model implementation and better feature utilisation. We gained important lessons from this project: namely, the power of intelligent dimensionality reduction and feature engineering to improve not only tractability of a model, but model feasibility and performance.

References

Charles R. Harris et. al. Array programming with NumPy. *Nature*, 585(7825):357–362, September 2020. doi: 10.1038/s41586-020-2649-2. URL `https://doi.org/10.1038/s41586-020-2649-2`.

Hadley Wickham et. al. Welcome to the tidyverse. *Journal of Open Source Software*, 4(43):1686, 2019. doi: 10.21105/joss.01686.

J. D. Hunter. Matplotlib: A 2d graphics environment. *Computing in Science & Engineering*, 9(3):90–95, 2007. doi: 10.1109/MCSE.2007.55.

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

Efstathios Stamatatos. A survey of modern authorship attribution methods. *Journal of the American Society for Information Science and Technology*, 60(3):538–556, 2009. doi: https://doi.org/10.1002/asi.21001. URL https://onlinelibrary.wiley.com/doi/abs/10.1002/asi.21001.