COMP90051 Project 2 Report

Group 43: Abhishek Dagar (1338713), Sahil Sinha (1231481), Pavith Samarakoon (1297058)

1. Introduction

In this project we try to handle the Authorship Attribution(AA) problem. It deals with predicting which authors among the set of 100 prolific authors are involved in writing the paper. For this problem we are provided with a training set of 25.8k papers with features like 'authors', 'year', 'venue', 'title',and 'abstract'. All of these attributes are encoded and are represented by numerical values. Apart from this, we also have another 800 papers as test data for which AA task needs to be done.

To implement this task we chose classical approaches using Naive Bayes and Linear SVC, and also chose an Artificial Neural Network (ANN) approach by building a Multilayer Perceptron model. We also discuss the results and limitation of the models, and why we chose different feature engineering methods for our task.

2. Feature Engineering

2.1 One-hot encoding

The current dataset has integer encoding performed, which is not good for machine learning(ML) models as it creates natural order relationships among each other. Also, it makes ML models assume that higher numbers have higher value. To remove these false relationships, one-hot encoding is performed on all the features.

2.2 Word2Vec

Word2Vec aims to learn the semantic/syntactic meaning of words by using a continuous projection layer to predict the context (nearby words) given a word (Skip-Gram model) or predict a word given the context (CBOW). This can create more informative embeddings to be used in ML models, with similar vectors/embeddings for similar words.[6]

2.3 TF-IDF

The term TF-IDF stands for term frequency-inverse document frequency. It is a well-recognized method to evaluate the importance of a word in a document. It is applied to 'abstract' and 'title' features to remove insignificant words and add importance to the words which gives more relevance to that particular paper. The features are the product of the frequency of each word in the document and the log of inverse document frequency for the word.

3. Problem transformations (Label powerset (LP) vs alternatives)

Classification models often predict single labels/classes for a given example, to apply such classification models to multi-label classification, we chose to transform the problem such that it could be modelled by a multi-class approach. One approach is binary relevance (BR), which applies a binary classifier to each class/label in the training set. Unfortunately, BR fails to capture potential correlations between labels as each label prediction is independent.[5] Classifier chains (CC) and LP are alternative problem transformations which attempt to learn label correlations. CC modifies the approach of BR by combining the binary classifiers into a chain, with each classifier using the predictions of earlier classifiers as an input. However, the model performance is highly dependent on the order of labels[5]; with L! possible orderings for L labels. LP performs multi-class classification on all label-sets present in the training set; this enables some of the correlation/dependencies between labels to be captured by the model by combining instances of multiple labels into a class.[5] A drawback to LP is the tendency for the number of classes to increase as the number of examples in the training set increases, and it is also unable to predict label-sets not seen in the training set; BR and CC are able to predict unseen combinations.

When applied to this problem, LP was able significantly outperform BR and CC, with validation f1 scores (using *4.2 Naive Bayes Model*) of 0.680, 0.044, and 0.063 respectively. This large difference in performance may be due to BR's inability to learn label correlations, and CC's dependence on the order

of classifiers in its chain (a different chain order may have performed better). It seems that even with its inability to predict unseen label combinations, LP was able to perform well by using the label relationships present in the training set, possibly due to authors often working with the same partners.

4. Experimentation and Discussion

4.1 Multilayer Perceptron - Initial Approach

As the name suggests, Multilayer Perceptron (MLP) consists of more than one layer of neurons. It is a fully connected class of ANN. A MLP model has 3 types of layers: input layer, hidden layers and output layer. It uses “backpropagation”[3] training algorithm to update weights and train the model.

We applied one-hot encoding to all the features to convert them to uniform n-dimensional feature space. In addition we applied CountVectorizer and Word2Vec embedding on Abstract and Title features so as to learn the relationship between different words.

Once feature engineering was completed, we split the data into training and validation data sets with a validation set consisting of 1% of the total training dataset. Then, we built a 3 layered MLP model with 22625 nodes at Input layer, 512 nodes in the hidden layer and 101 nodes at output layer. To train the model we utilised Adam’s optimizer, used binary cross entropy loss function as it had multiple binary labels at output layer and used early stopping criteria with patience 20 to stop our model from overfitting.

The approach chosen here, became a good benchmark and a starting point for comparing our models. Using this approach we were able to achieve only a 0.408 F1 score in Kaggle. It did not perform well as we were not able to handle multi-label classification problems and treated it as a multi-class problem. Here, we were able to get only a single label as an output, which was decided by the maximum likelihood of the label for that particular input. Also, using One-hot encoding resulted in multi-collinearity among various variables resulting in lowering of model’s accuracy.

4.2 Naive Bayes Model - Final Approach 1

Naive Bayes (NB) classifiers are based on Bayes Theorem, with the assumption of independence between features/predictors (X) given the class/label (C) to simplify the estimation of P(X|C), which can otherwise be difficult for high dimensional feature spaces. The assumption of independence is often not true in practice, however, NB classifiers have been effective for multi-class classification, especially when features are completely independent or functionally dependent.[1]

To obtain a fixed dimensional feature/input space suitable for the NB classifier, Tf-Idf vectorisation was applied to the text data. Multi-class classifiers predict a single class for the given data, however, this AA problem requires multi-label classification. So, we used label powerset (LP) to transform this problem into a multi-class classification, this approach was discussed further in ‘*3. Problem transformations*’.

As the training data was very imbalanced, we sampled data with ‘-1’ labels in the training data to produce a training dataset with 20% ‘-1’ labels. With imbalanced datasets some classes may have very few examples, complement NB was used to overcome this limitation by calculating the probability for the complement of each class. Thus transforming the NB classification problem with limited examples for classes into one with many examples for each (complement) class.[4] Multinomial NB was also tested with LP, however, as expected it performed worse on the validation set than complement NB (f1: 0.195 vs 0.680), likely due to the small number of samples available for some classes. Complement NB combined with LP was able to produce significant improvements over the MLP model, with an F1 score of ~0.526 on the Kaggle test set, likely due to this model’s ability to use correlations/dependencies between labels to predict multiple labels for a given example.

4.3 Linear Support Vector Classifier - Final Approach 2

SVM, Support Vector Machine, is basically a linear model which is used for the problems mainly related to regression and classification. A Support Vector Machine model intends to solve both linear as well as non-linear problems, and has shown good results and performance for quite many practical problem statements. At its core, the algorithm for the SVM model creates a line or a hyperplane which then separates the data further into classes.

As we already know, the linear SVC model tends to support both sparse and dense inputs given to it, and handles multiclass support as per a on-vs-the-rest scheme. So, we have leveraged Label powerset's multi-class classification scheme and then combined it with Linear SVC so as to produce the best results. Moreover, we can note that Linear SVC is faster when it comes to converging large numbers of samples. Plus, it also minimises the squared hinge loss as well as penalises the size of the bias. With this benefit, we were able to get a f1 score of ~0.53. But after a few more tweaks, we could achieve the final score of ~0.555. We were able to do this simply by adjusting the data of non-prolific authors ('-1' label) from 25% to 17%.

5. Conclusion

In this project, we tried to solve the Authorship Attribution problem. We designed and developed three models, two using traditional ML algorithms- Naive Bayes and LinearSVC, and one using MLP. We tried multiple problem transformation approaches like Binary Relevance, Classifier Chain and Label Powerset to transform our problem from multi-label classification to multi-class classification. We were able to effectively utilise these for our traditional ML algorithms and were able to compare the results, with Label Powerset significantly outperforming the other methods (0.6 higher validation F1 score when tested on Naive Bayes model). For the MLP implementation we tried several feature engineering approaches like One-hot encoding, Word2Vec and TF-IDF. After our experiments we found our classical approach trumped the ANN(F1 score: 0.408) by far and LinearSVC(F1 score: 0.557) gave us the best results with Naive Bayes(F1 score: 0.526) following it at close second.

References:

[1] Rish, Irina. T.J. Watson Research Center (2001). An Empirical Study of the Naïve Bayes Classifier. IJCAI 2001 Workshop on empirical methods in artificial Intelligence.
[2] Nooney, K. (2019) Deep dive into multi-label classification..! (with detailed case study), Medium. Towards Data Science. Available at: https://towardsdatascience.com/journey-to-the-center-of-multi-label-classification-384c40229bff
[3] David E. Rumelhart; James L. McClelland, "Learning Internal Representations by Error Propagation," in Parallel Distributed Processing: Explorations in the Microstructure of Cognition: Foundations , MIT Press, 1987, pp.318-362.
[4] Rennie, Jason & Shih, Lawrence & Teevan, Jaime & Karger, David. (2003). Tackling the Poor Assumptions of Naive Bayes Text Classifiers. Proceedings of the Twentieth International Conference on Machine Learning. 41.
[5] Bogatinovski, J. et al. (2022) "Comprehensive comparative study of multi-label classification methods," Expert Systems with Applications, 203, p. 117215.
[6] Mikolov, Tomas & Chen, Kai & Corrado, G.s & Dean, Jeffrey. (2013). Efficient Estimation of Word Representations in Vector Space. Proceedings of Workshop at ICLR. 2013.