COMP90051 Project 1: Author Attribution

Group 53

April 25, 2022

1 Introduction

The problem given is an author attribution task, which requires us to identify whether an author participated in the writing of a document. We are provided with 26108 papers to train on, each one recording the venue it was published at, the year it was published, a list of specific keywords in the title and abstract, and a list of the authors of the paper. The task is to determine whether a given target author is a true co-author of a paper when provided with the true co-authors, venue, year, and keywords.

Integers are used as IDs for all values in the data instances. As mentioned in the project specification, the list of keywords only records the presence of certain keywords, as such we assume there are no duplicates and no order within the list and thus treat is as a set (the same is assumed for the list of authors). Where $[n]_0 = \{0, 1, 2, ..., n\}$, we will use the following:

- the set of all authors, $A = [2301]_0$
- the set of all venues, $V = [469]_0 \cup \{\text{“ ”}\}$
- the set of all keywords, $K = [499]_0$
- a training instance with ID n provided in train.json is (v, y, A_n, K_n) where $v \in V, A_n \subseteq A, K_n \subseteq K$ and y is a year
- a test instance with ID n provided in test.json is (v, y, A_n, K_n, t) where $v \in V, A_n \subset A, K_n \subseteq K, t \in A \setminus A_n$ and y is a year

We treated the authors as our labels in training, and since each training instance can have multiple authors, for the initial preprocessing we replaced each training instance (v, y, A_n, K_n) in the following way: for each $t \in A_n$ we create a new training instance $(v, y, A_n \setminus \{t\}, K_n, t)$, which is the same format as the test data.

2 Final approach

2.1 Binary representation

The training instances, even after our initial preprocessing, cannot be used as is due to them containing sets for the authors and keywords which are incompatible with most machine learning techniques. A simple solution would be to replace each possible set with an integer to represent it, however this would make it rather difficult for the model to notice trends between papers with similar but not identical coauthors without a carefully defined mapping function. This would also clearly cause an overflow since there are $2^{|A|}$ possible sets for A_n. The first solution we opted for was to represent A_n and K_n as a sequence of 0s and 1s in the following manner: given a set of coauthors A_n, we represent it as $a_n = (a_{n,0}, a_{n,1}, a_{n,2}, ..., a_{n,2301})$ where $a_{n,i} = 1$ iff $i \in A_n$, otherwise $a_{n,i} = 0$ (the same method was applied to the set of keywords too). We then joined the binary representations of the coauthors and keywords together with the publication year.

We considered ignoring the venue at which the paper was published since all papers without specific venues would be treated the same which we believed to be undesirable, however decided to include

it since results were good and the number of papers with no venue was relatively small (1795 out of 26108 papers). As such our final mapping was:

$$\mathcal{F}((v, y, A_n, K_n, t)) = (v, y, a_{n,0}, a_{n,1}, a_{n,2}, ..., a_{n,2301}, k_{n,0}, k_{n,1}, k_{n,2}, ..., k_{n,499})$$

This representation was tested and abandoned for our initial models (Naive Bayes and support vector machine) due to the size of each training instance after the mapping being too large for them to handle. Each instance now had $1 + |A| + |K| = 2803$ features, and with almost 50000 training instances after the initial preprocessing, the amount of time it took for models to train was too long for us to effectively use. However, our final model which used a neural network showed reasonable training time with this representation and, as such, we chose to use it for that specific model.

2.2 Model

For our chosen final approach, we used neural networks to optimise a multiclass logistic regression. PyTorch enabled us to efficiently perform differentiation on matrices, making it possible to train over a large feature space. Additionally, because training is implemented using stochastic gradient descent, training is much quicker since it does not calculate the true gradient from the entire dataset and instead uses an estimate.

The final approach uses all papers in the training set using a binary representation (see above) for the keywords and authors. Our initial attempt gave an AUC score of 0.873, however, our features were not normalised and some of our predicted probabilities were greater than 1. After normalising the venue and features values so values lay in a range of $[0, 1]$ our final AUC score was 0.881.

3 Alternatives considered

3.1 Padded representation

Inspection of the training data found that, while there may be over 2000 authors and 500 keywords in the entire dataset, neither the number of authors nor the number of keywords for each paper were anywhere near that amount. Thus we defined a new mapping as follows:

$$\mathcal{F}((v, y, A_n, K_n, t)) = (y, a_{n,0}, a_{n,1}, ..., a_{n,m_a}, k_{n,0}, k_{n,1}, ..., k_{n,m_k})$$

where m_a and m_k are the max number of authors and keywords in any given training instance respectively. The values of $a_{n,i}$ and $k_{n,i}$ were then simply taken as the values given in A_n and K_n respectively. This greatly reduced the number of features down to $1 + m_a + m_k$ which we found to be $1 + 9 + 109 = 119$ for the provided training set. Any instances that did not have the max number of authors/keywords were padded with the target author t and the first keyword in K_n. This representation was never tested by itself as we believed it too simplistic and naive. We attempted to modify it a little further as described below.

3.1.1 Coauthor and keyword matrices

Instead of simply taking the padded sets of coauthors and keywords, we tried to order them such that coauthors and keywords more strongly associated with the target author would be further to the left, hoping that our model would be able to recognise these features as more significant. To do this, while preprocessing, we simply looked through the training instances and constructed the following matrices which were used to lookup the required priorities for each author/keyword:

$$P_a = \begin{pmatrix} p_{0,0} & p_{0,1} & \cdots & p_{0,2301} \\ p_{1,0} & \ddots & & \vdots \\ \vdots & & \ddots & \vdots \\ p_{2301,0} & \cdots & \cdots & p_{2301,2301} \end{pmatrix}, P_k = \begin{pmatrix} p_{0,0} & p_{0,1} & \cdots & p_{0,500} \\ p_{1,0} & \ddots & & \vdots \\ \vdots & & \ddots & \vdots \\ p_{2301,0} & \cdots & \cdots & p_{2301,500} \end{pmatrix},$$

Here, the value $p_{i,j}$ represents the number of papers author i has written with author j in P_a and the number of papers author i has written with keyword j in P_k.

3.2 Prediction based on research groups

Among the authors in the provided data, we hypothesised that there might be research groups where group members tend to publish papers together more often. Mapping the feature authors into research groups was a way to reduce the feature dimensionality and hence improve the speed of model training. From the 26108 papers we found that they were written by over 9000 different author groups. By removing the groups which are subsets of other groups, we found 4348 research groups. We used this information as the base of one of our first naive frequentist model attempts, which assigned a high probability to the target author of the test paper when the target author and any co-author in the co-author list are in a research group. This model only makes use of a single feature and achieved an AUC score higher than 0.667.

This served as a good start to our project but we believed that we could achieve better scores with improved models and abandoned this approach.

3.3 Naive Bayes

The main machine learning model we initially experimented with was Naive Bayes. We believed that since it was a probabilistic classifier it was suited to the task which required probability predictions instead of a yes/no prediction.

Since Naive Bayes calculates the probability $P(t = n|venue, year, coauthors, keywords)$ for all target authors, we could simply predict these probabilities for the given test input and use the probability of the target author as our prediction. The model performance was rather poor but served as a baseline for other models, giving scores of 0.504 with the padded representation (see section 3.1) and 0.667 with the research group representation (see section 3.2).

We believe the model did not perform very well mainly due to the feature independence assumption not holding. There is likely some kind of correlation between authors and keywords since an author is more likely to write about the same topics, and thus have papers with similar keywords.

3.4 Multi-layered perceptron

We tried using a multi-layered perceptron to train our data. To process the data we first divided the author into groups. Using each group as labels and the year, venue, coauthors, and keywords as instance features to train the MLP model. Train each model for a maximum of 250 iterations and finally, we get multiple classifiers, one for each group of authors. When testing, for each test instance we decide which group the target author is in, and use the corresponding classifier to determine whether the target is a true co-author or not. In the end, we get a score of 0.865 with 16 author groups. The main reason this approach was not used is because the score cannot be reliably improved by simply increasing the iterations of training or adding more classifiers. This method also takes a long time to train which makes it hard to efficiently test and improve, and also ignores any potential correlation between coauthors and the target.