An Analysis of Dimensionality Reduction Techniques in Regards to Authorship Classification

Joshua Shapiro, Ishan Sharma
Department of Computer Science
The George Washington University
Washington, DC

Abstract

The purpose of this project is as follows: Given multiple bodies of text written by different authors, we want to determine with the highest accuracy which authors wrote which texts. The end goal of this analysis is to discover the best feature selection technique and classifier pair that yields the highest accuracy for authorship classification.

1 Introduction

Authorship Classification has been a long studied problem in the domain of linguistics. Even before the present day field of Computational Linguistics, the idea that authors wrote with a subconscious style existed. In the 1400s a man by the name of Lorenzo Valla used this idea to prove that the famous Donation of Constantine was a forgery. By looking at the grammatical structure, vocabulary, and tone of the piece, Valla determined that Constantine was not the true author of the letter. Today, similar problems of authorship classification can be solved using computational approaches, however the techniques vary tremendously with regard to the features used to identify the author. This project aims to determine which feature selection processes yield the highest accuracy for authorship classification. The baseline accuracy will be based on trigram models of the training data.

We begin by describing the dataset chosen and what makes it a good option for this project. Then we explain how the data was preprocessed in preparation for analysis. Next the manual feature selection approaches are covered in detail. We move on to explain the automated dimensionality reduction techniques used on the feature sets as well, and finally analyze the different classifiers used. Finally we discuss the accuracy results of the classifiers and feature selection techniques on the dataset.

2 Data

To test our approach for authorship classification, we used Reuter_50_50 Data Set. It contains 50 training documents and 50 test documents for each of the 50 authors, for a total of 2,500 training documents and 2,500 test documents. This dataset is extremely useful for authorship analysis since all of these documents focus on news. This means that topic of writing cannot be used to determine authorship, and more importantly the training data for each author will be relatively homogeneous.

3 Feature Extraction

The data from Reuter_50_50 is used to create features and instances suitable for classification.. Our project extracts the features for each condition. In our algorithm, we made a matrix with article number as rows and features as columns. In the following sections, I will describe these processes of feature extraction.

4 Manual Feature Selection

Feature selection is a procedure in machine learning to select a subset of features that produces better model for given set. Hence, after selection has taken place, the data set should still have most of the important information present. In fact, a good feature selection technique should be able to detect and ignore noisy and misleading features. The result of this is that the data set quality might even increase after selection.

4.1 Bag of Words

The idea of Bag of Words (BoW) is to count how many times a word appears in a document. Those word counts allow us to compare documents and gauge their similarities for applications like search, document classification, in our case authorship classification. In other words, The Bag of Words model learns a vocabulary from all of the documents, then models each document by counting the number of times each word appears.

In our algorithm, we used Scikit-learn which provides methods for the most common ways to extract numerical features from text content[3]:

- tokenizing strings and giving an integer id for each possible token, for instance by using white-spaces and punctuation as token separators.
- counting the occurrences of tokens in each document.
- normalizing and weighting with diminishing importance tokens that occur in the majority of samples / documents.

In this scheme, features and samples are defined as follows:

- each individual token occurrence frequency (normalized or not) is treated as a feature.
- the vector of all the token frequencies for a given document is considered a multivariate sample.

A corpus of documents can thus be represented by a matrix with one row per document and one column per token (e.g. word) occurring in the corpus.
The general algorithm for bag of words feature selection using Scikit-learn is as follows[3]:

```
from sklearn.feature_extraction.text import CountVectorizer
vectorizer = CountVectorizer(min_df=1)
X = vectorizer.fit_transform(corpus)
from sklearn.feature_extraction.text import TfidfVectorizer
vectorizer = TfidfVectorizer(min_df=1)
vectorizer.fit_transform(corpus)
```

Through CountVectorizer we were able to perform vectorization, which is process of converting a collection of text document into numerical feature vectors. Tf-idf is used in order to re-weight the count features into floating point values, which is suitable for classifier usage.

4.2 Stylometric Features

The statistical analysis of style, stylometry, is based on the assumption that every author's style has certain features being accessible to conscious manipulation. Therefore they are considered to provide a reliable basis for the identification of an author. In our algorithm, we used following features:

Number of sentences	Number of tokens
Average sentence length	Average token length
Frequency of pronouns	Frequency of semicolons
Frequency of exclamation marks	Frequency of periods
Frequency of question marks	Frequency of commas

Fig. 1

4.3 Part Of Speech (POS) Tags:

The basis of our method is extremely simple: run the texts through a parts of speech tagger and label each corresponding part with a symbol. In our algorithm, we used frequency of part of speech tags. It was built like the way we built Bag of Words feature using CountVectorizer.

5 Automated Dimensionality Reduction

In addition to selecting subsets of features as described above, the problem of authorship classification is well suited for dimensionality reduction techniques. While in some classification problems it is useful or necessary to know which features are contributing to the accuracy of the classification, it does not matter as much with determining authorship. For this reason, dimensionality reduction techniques that transform data in high dimensional space to lower dimensional space can be used on our project. Below are descriptions of the dimensionality reduction algorithms used.

5.1 PCA Decomposition

The idea of Principle Component Analysis is to find the principal components of data. What this means is that PCA decomposition is able to find the underlying structure in the data by looking for the directions of data with the most variance. These principal components can be calculated based on eigenvectors and eigenvalues. For each eigenvector and eigenvalue pair, the vector specifies the direction and the value specifies the amount of variance. Therefore, the most significant principal component is the one whose direction corresponds to the largest eigenvalue. As there is an eigenvalue/eigenvector pair for each column of the data, an MxN dimensional dataset will have N principal components, equal to the number of features for each datapoint.

Typically, not all principal components will be made from substantial eigenvalues. That is, the first few components will show large variances in the data, while the remaining components show very little variance, or none at all. For this reason, it is common to simply drop the less important principal components. Through this removal, the feature set dimensions can be reduced. In this project, we are testing PCA Decomposition keeping 10%, 25%, and 50% of the initial feature size.

5.2 Feature Agglomeration

The goal of Feature Agglomeration is to identify features that behave similarly, and group them together into clusters. Then, from each cluster one feature is selected as a representative. This group of representatives is then a subset of the initial features, and therefore reduces the dimensionality of the original feature set. Unlike PCA, the actual features are not modified in any way, so the reduced feature set can show which features are most significant in authorship attribution.

The general algorithm of feature agglomeration is as follows. Each individual feature starts in its own cluster. Then clusters that are similar are merged. The clusters are continually merged until only one cluster remains, and then different levels of the tree structure created can be used for feature pruning. This similarity metric used in part to identify which clusters should be merged can be measured in a variety of ways, but for our example it is the euclidean distance between values. Once the similarity values between clusters are calculated, a linkage criteria determines which clusters combine. For the implemented algorithm, Ward's linkage criteria is used. The goal for this criteria is to minimize variance within each cluster, and uses the euclidean distance measure as part of its calculation. As feature agglomeration is a form of hierarchical clustering, it determines the cluster merges in a greedy manner, and the time complexity is $O(n^2 \log(n))$, where n is the number of initial features. Just as we did for PCA, we are testing feature agglomeration and keeping 10%, 25%, and 50% of the initial feature size.

5.3 Random Projection

Unfortunately both PCA and feature agglomeration can be time intensive to run if the number of features is large. Because of this, the final dimensionality reduction strategy chosen is one that performs extremely quickly. The idea of random projection is based on the Johnson-Lindenstrauss lemma. This states that data points in a vector space of sufficiently high dimension can be projected into a suitable lower dimensional space that still preserves the pairwise distances between points. This distance preservation is important, as it means that all classifiers that depend on distances between points (KNN, perceptron, etc) should perform with close to the same accuracy on the data projected to a lower dimensional space as it did on the original data.The general premise of random projection is to take the initial featureset matrix and multiply it by a randomly generated matrix that outputs a new matrix with fewer dimensions. There multiple ways to generate this random matrix, and we are testing two.

The first way is using a gaussian distribution. The components of the matrix are drawn from a gaussian distribution, and this ensures that there is spherical symmetry between the initial data and the dimension reduced data. Additionally, the rows of the matrix are orthogonal to each other, and the rows are unit length vectors. The second way is using a sparse random distribution. This guarantees similar quality to a gaussian distribution, but it is more memory efficient and allows faster computation of the projected data as it produces a sparse matrix. The components of the matrix come from the following:

$$\frac{-\sqrt{s}}{\sqrt{nComponents}} \quad \text{with probability } \frac{1}{2*s}$$

$$0 \qquad\qquad \text{with probability } 1-\frac{1}{s}$$

$$\frac{\sqrt{s}}{\sqrt{nComponents}} \qquad \text{with probability } \frac{1}{2*s}$$

$$\ldots\text{where } s = \frac{1}{density\ the\ components\ of\ the\ random\ matrix\ are\ drawn\ from}$$

$$\text{and } nComponents\ =\ number\ of\ reduced\ dimensions$$

For each type of random matrix generation, nComponents is tested at 10%, 25%, and 50% of the initial feature size. Additionally, the random projection algorithms allow nComponents to be set to auto, and the algorithms choose the required number of dimensions to reduce to in order to keep the quality of the embedding at 0.1 (lower number implies higher quality of embedding). Because of this, nComponents is also tested with "auto".

6 Classifier Analysis

6.1 Setup

Two types of classifiers were chosen for this authorship classification problem. We used support vector machines with a handful of kernels to represent discriminative learning and Naive Bayes with two different implementations to represent generative learning. As the focus of the project is with feature extraction and selection, the description of the classifiers used will be relatively brief. All classifiers were trained on the training data after it had been run through one of the feature selection techniques discussed above. This training data consists of 50 articles each written by 50 authors, for a total of 2,500 articles. Once the classifiers were created, they were evaluated on a validation set which consists of 25 articles each written by 50 authors, for a total of 1,250 articles. After the best feature selection/classifier pairs were identified, they were evaluated against the test set for final results. The test set is the same size as the validation set. The data in each set is mutually exclusive, and all sets have uniform distributions against the authors.

6.2 Kernel SVM

As all of the code was written in Python, we used Scikit Learn's built-in Kernel SVM function. For the function inputs we tried various kernels with different soft margin constraints (c). This enabled us to do a grid-like search over linear, polynomial degree 2, polynomial degree 3, and gaussian radial basis function kernels with soft margin constraints of 0.1, 1, and 10. All combinations of these parameters were run over all feature reduction methods. As this project is a multi class classification problem, the classifiers were built using a one versus rest model.

6.3 Naive Bayes

For our Naive Bayes classifier, we used Scikit Learn's built-in naive bayes algorithms. As there are no parameters to tune for the implementation of naive bayes, we tried two different types of algorithms. The first is the Gaussian Naive Bayes algorithm. In this implementation, the likelihood of the features is assumed to be Gaussian, and is calculated in the following way:

$$P(x_i|y)\ =\ \frac{1}{\sqrt{2\pi\sigma_y^2}}exp(-\frac{(x_1-\mu_y)^2}{2\sigma_y^2})$$

Where σ_y and μ_y are estimated using maximum likelihood

The second is the Multinomial Naive Bayes algorithm. This implementation is one of the classic naive bayes variants for text classification since most text classification problems can be modeled as a multinomial distribution. In this implementation, the probability of feature i appearing in a sample belonging to class y is calculated in the following way:

$$P(x_i|y) = \frac{N_{yi}+\alpha}{N_y+\alpha n}$$

Where N_{yi} is the number of times feature i appears in a sample of class y, N_y is the total count of all features for class y, and $\alpha = 1$ for Laplace smoothing

Both naive bayes classifiers were run over all feature reduction methods discussed in sections 3 and 4.

7 Conclusions

In our experiment we were able to get the 67.96% accuracy for SVM (Support Vector Machine) classifier and PCA (Principle Component Analysis). In fig. 2, each classifier is a section of a bar graph and each dimensionality reduction technique a bar of each section. We analysed the most accurate value for each classification algorithm.

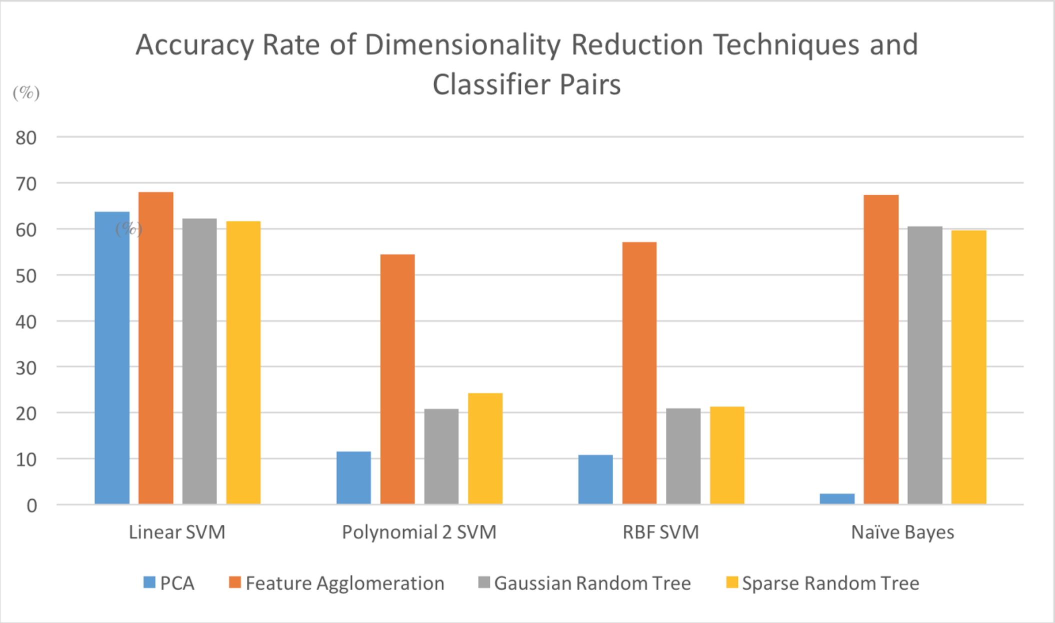

Fig. 2

References

[1] Peng, F.; Schuurmans, D.; Keselj, V; Wang, S. (2003). Language Independent Authorship Attribution using Character Level Language Models. Proceedings of the 10th Conference of the European Chapter of the Association for Computational Linguistics, Budapest, 267--274

[2] ZhiLiu, Reuter_50_50 Data Set. Retrieved November 15, 2016, from http://archive.ics.uci.edu/ml/datasets/Reuter_50_50

[3](n.d.). Retrieved December 19, 2016, from http://scikit-learn.org/stable/tutorial/

[4] Elayidom, S., Jose, C., Puthussery, A., & Sasi, N. K. (2013, September 5). TEXT CLASSIFICATION FOR AUTHORSHIP ATTRIBUTION ANALYSIS. 1-8. Retrieved November
14, 2016, from http://airccse.org/journal/acij/papers/4513acij01.pdf

[5] AICBT, Authorship Attribution with Python. Retrieved November 8, 2016, from http://www.aicbt.com/authorship-attribution/