Blog Authorship Classification

Jaime Pacheco

Abstract

Classification is a central task within the field of NLP. In this paper we examine how classification can be used to assign shared blog authorship among different groups of texts.

We explore an unsupervised method of classification, which relies on the bag of words model to obtain blog vector representations. We show various methods of using these vectors to determine author clusters, including cosine similarity and PCA reductions.

We show that blog authorship classification is a difficult task, and that producing blog clusters by author given our current methodology is not reliable. Topic clustering appears to be much more successful, but still has room to be optimized.

1 Introduction

The main goal of unsupervised blog authorship classification is to take an unlabeled group of texts, and to arrange them in different clusters based on their author.

1.1 Data set

Within this project, we will use the Blog Authorship Corpus (Jonathan Schler and Pennebaker, 2006). This is a data set containing 681,288 posts from 19,320 different authors recorded on or before August 2004. Each entry contains the text data, the topic, the date it was written, an id corresponding to the author, and several other statistics about the author. The data set is so large that it contains over 140 million words. This is far too much data to work with in a single experiment, so we must sample the data effectively. For our purposes, sampling the N first entries is sufficient. The data is already clustered by author, and we wish to maintain those clusters in our data, such that we can attempt to recreate them experimentally.

1.2 Problem

In this project, I will be exploring is whether I can replicate similar results to what related works have found regarding authorship classification. Namely, can I classify blogs based on their author by using unsupervised methods akin to what will be discussed in the related works? How feasible is it to create clusters of the data based on their authors? If it is not reliable based on author, can I create clusters based on topic? How can I optimize these strategies to get the best results given the data set?

2 Related Works

Authorship Classification is a highly sought after goal within the field of NLP. Unsupervised classification is a particularly compelling area of research, as it requires less human work in a field where data sets can be particularly large. There are various examples of experimentation in this field that we will explore, many of which introduce unique approaches to the problem.

2.1 Similarity

When it comes to unsupervised authorship classification, similarity-based approaches are usually the go to. These are effective because it allows us to easily measure the similarity of two documents, provided they can be represented as a vector. Much of the complexity of this approach comes from how we use vectors to represent our documents.

Most embeddings are obtained by some form of the bag of words model, where the frequency of each word is a feature of each document. Making use of other models like WordNet, Glove, and Word2vec proves to be very effective (Zied Haj-Yahia and Deleris, 2019). These models allow us to obtain synonym relationships between words, and to keep track of words that are semantically similar.

Another extremely useful tool used is Lbl2Vec (Tim Schopf and Matthes, 2021), as it jointly embeds word, document, and label representations. It

does this by first using Doc2vec to obtain word and document representations. From there, we can use cosine similarity to find a set of the most similar document representations, using the average of the keyword label representations for each class. We can generate label vectors for the classes using the average of the document representations. Documents end up assigned to the classes with the highest cosine similarity among the label and document vectors, thereby succeeding in tasks like authorship classification. This method can be modified slightly, where instead of Doc2vec, transformer models like SimCSE and SBERT can be used to create text representations (Tim Schopf and Matthes, 2022), (Stammbach and Ash, 2022). This results in a slight improvement over the traditional model.

2.2 Dimensionality Reduction

In performing blog authorship classification, one of the most common issues is having data that is way too large. Word frequencies are often used as features, which can lead to having extremely large vectors representing individual documents. These large sizes make it difficult to work with the data. Thankfully, there are various techniques that can be used to reduce the size of these vectors and the number of features considered, without losing too much information. Common examples include Principal Component Analysis (PCA) and Latent Semantic Analysis (LSA). Both of these methods work by projecting the data onto a smaller subspace that maintains the fit of the original data. In terms of computation, these methods require computing the singular values of the data matrix. Both of these techniques can be extremely useful, and a paper by B. Ljungberg seeks to determine which of these two methods is optimal in a blog authorship classification setting (Ljungberg). The paper compares these strategies in an experiment on a group of texts represented with a bag of words model. It ultimately determines that PCA outperforms LSA.

3 Methods

3.1 Gensim

The bulk of the methods implemented in this project rely on Gensim, a python library built for topic modelling, document indexing and similarity retrieval. It includes various algorithms for popular tasks, and it is memory efficient (gen, 2022).

3.2 Pre-processing

In order to work with the Blog Authorship Corpus, we must preprocess the data. The first step is to strip each blog of all its tags, punctuation (replaced with spaces), non-alphabetic characters, and digits. We also remove all of the stop words (i.e. 'a', 'the', etc.), and all words of length two or less. Finally, we convert the entire document to lowercase and stem the remaining tokens. It uses the porter-stemmer, which is an algorithm that removes common morphemes from the ends of words (Porter, 2006). This helps remove inconsequential variances from texts, and allows us to more easily observe similarities between texts.

3.3 Bag of Words

Once all the data has been pre-processed, we can introduce a bag of words model. Each blog is considered to be a bag of words, which means that while the frequency of words is relevant, their order is not. We implement this model by creating a dictionary that includes all unique words across the entirety of the blog data. For each blog, we can represent its text data as a vector including the frequency of each word from the dictionary. We can therefore maintain a matrix of all the blog data.

3.4 Term Frequency - Inverse Document Frequency (TF-IDF)

Once our blog vectors are created, we can run TF-IDF to balance out the influence of words. TF-IDF looks at term frequency, the amount of times a word appears in a document, and inverse document frequency, the inverse of the number of documents that contain a given word. Scaling these parameters and taking the product helps assign a weight to each word that is representative of its role in the document. This method is crucial because it assigns higher weight to less common words. Uncommon words can be the key to differentiating texts from one another, and so we must make sure that these are properly accounted for.

3.5 Similarity

In order to accurately determine relationships between blogs, we must be able to compare them to each other. Cosine similarity is the optimal method for this, as we can now pass in the weighted vectors that represent each document as our input. Cosine similarity is defined as $cos(\theta) = \frac{x \cdot y}{||x||||y||}$. In this case, θ represents the angle between the two doc-

ument vectors. Small angles between them will result in high measures of similarity. This is an effective measure as it is resistant to the length of the documents. Gensim has built in methods that take in an input vector, and can compute the pairwise similarity of that vector with every row of a given matrix (gen, 2016).

3.6 Principal Component Analysis (PCA)

The final method that we incorporate is PCA. The matrix we have that represents our entire dataset is likely to be very large, as it has a row for each document, and its number of columns is equivalent to the number of unique words across all documents. PCA serves to reduce this size into something that we can digest. It allows us to reduce our documents to lower dimensions via linear transformations, while still maintaining sufficient information about the document. For our purposes, we can reduce the dimensionality of our documents to 2, in order to plot these vectors in the x-y plane. By plotting each document we can observe the clusters and relationships between them directly. (Fontana, 2020)

4 Results

4.1 Similarity Evaluation

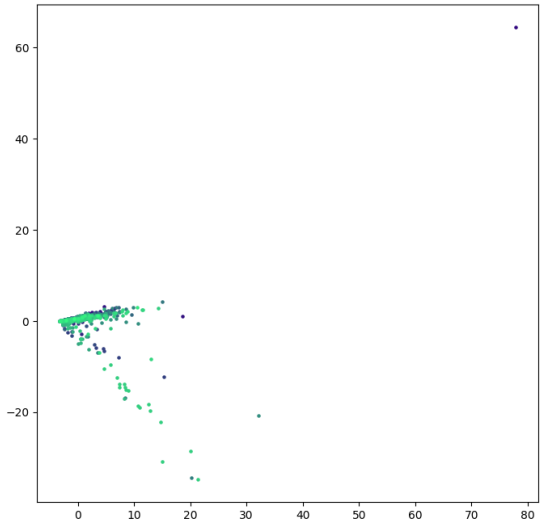
Once we have produced vector representations of the documents, we can apply cosine similarity to any given pair to view how similar they are. We'd like to use this metric to get an overall perspective on how likely we are to correctly cluster documents by author.

One way to do this is to take a given blog, and output the blogs with which it has highest similarity. In the best case these blogs should be other blogs written by the same author. However, it is likely that other blogs will present as well, in some cases having higher scores than those written by the same author. One way we can experiment with this is as follows:

Given an author A who wrote a set of N blogs, compute the N highest cosine similarity scores. The percentage of these scores that have the same author defines the accuracy of our method.

Since each blog written by an author can still be unique in terms of content, it's unlikely that it will match the exact same similarities as another blog by the same person. Thus we can circumvent this variance by computing the accuracy for each blog written by an author, and averaging them.

Figure 1: A plot of 1000 blog vectors obtained from the methods in section 3. The color of each point corresponds to document's author.



In the table below, we compute the accuracy for a few of the authors from the dataset. Additionally, we compute the accuracy for all authors across our sample. In this experiment we sampled the first 4000 blog entries from our set.

Table 1:

Author	Accuracy
2059027	0.250
3581210	0.498
3539003	0.125
4172416	0.5
3668238	0.29
All Authors	0.311

Additionally, the standard deviation of the accuracy of all authors comes to 0.145.

4.2 PCA Plot

After running the PCA described in method 3.6, we can obtain the following plots (figures 1, 2, and 3). These plots allow us to directly observe the clusters of blogs, and which are the most similar. Each plot is run on a sample of the first 1000 blog entries from the data set.

From these three figures we can observe some relation between certain blogs, and some association between those with shared authorship. Figure 3, does a particularly good job of this, as it isolates the problem to just arranging the vectors of one author. However, even within just one author

Figure 2: A plot of 1000 blog vectors obtained from the methods in section 3. The color of each point corresponds to documents' topic.

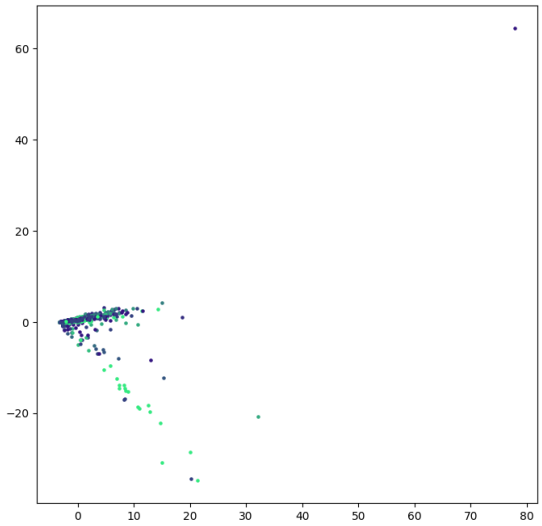
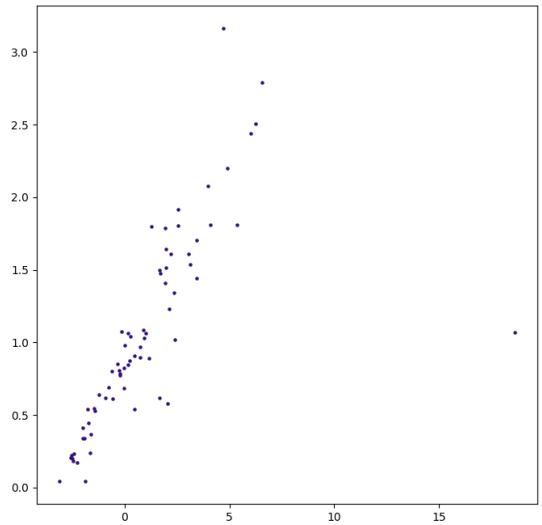


Figure 3: A plot of the blog vectors obtained specifically for blogs written by the author with id '3581210'.



we can see a pretty large variance across the data. Some individual pairs can be located nearby, but some points spread far and wide across the graph. The further away these points are, the larger the angle they create with respect to the origin, and therefore the cosine similarity is reduced. Thus it's somewhat hard to see why we would consider points like those to be written by the same author, without prior knowledge.

Figures 1 and 2 show a bigger picture, having all 1000 sampled entries included. Here it's more evident that points having the same color does not necessarily mean that they will be closer together. Figure 2 does seem to be a slight improvement over figure 1, as we see less disparity between points of the same color. However, none of these plots seem to produce 100% convincing clusters.

5 Conclusion

From the data we've obtained, it's evident that while authorship classification is possible, there is a fair amount of error as well. The similarity data shows that our method is not very precise, being that the average accuracy across all sampled authors is 0.311. On average, we are retrieving more blogs from different authors than we are from the same author on a given post. There is also clearly a high amount of variance among the data, as can be seen with the fact that the standard deviation across all authors is 0.145. This is reasonable, as whether or not a certain author wrote an article is not the only variable at play here. The topic of the blog has everything to do with this, and any given blog could be related to a popular topic, which could draw in more blogs from different authors.

The influence of topic was explored in the PCA experiments. They showed that while authorship clustering is a mess, topic clustering can clear things up slightly, although perhaps not by much. Both show some sense of clustering, but also a relatively high degree of variance among elements of the same class.

The trends observed in these experiments are reasonable given our methodology. Most blogs in the data set are not very long, and many of them reuse much of the same vocabulary. Often times, the difference between blogs comes down to one or two key words. These key words, words that are somewhat unique to that specific blog, are often more correlated with the topic than the author. For example, blogs related to writing html will use

the word html, and thus match with other blogs that discuss html, regardless of if the same author wrote about other, non-related to html topics. The influence of these key words is certainly increased by the implementation of TF-IDF, which as we recall is used to assign higher weight to lesser used words. In the future, we could explore methods that lessen or remove TF-IDF to avoid this effect. It could also be that with larger samples of data, our results improve. The related works section provides a good perspective on some higher level techniques that could give us better results. Investigating the effectiveness of these technqiues is a great future direction for this project.

References

2016. gensim.similarities.matrixsimilarity.

2022. gensim 4.2.0 project description.

Eleonora Fontana. 2020. Bow + tf-idf in python for unsupervised learning task.

Shlomo Argamon Jonathan Schler, Moshe Koppel and James Pennebaker. 2006. Effects of age and gender on blogging.

Benjamin Fayyazuddin Ljungberg. Dimensionality reduction for bag-of-words models: Pca vs lsa.

Martin Porter. 2006. Porter stemming algorithm.

Dominik Stammbach and Elliott Ash. 2022. Docscan: Unsupervised text classification via learning from neighbors.

Daniel Braun Tim Schopf and Florian Matthes. 2021. Lbl2vec: An embedding-based approach for unsupervised document retrieval on predefined topics.

Daniel Braun Tim Schopf and Florian Matthes. 2022. Evaluating unsupervised text classification: Zero-shot and similarity-based approaches.

Adrien Sieg Zied Haj-Yahia and Lea A. Deleris. 2019. Towards unsupervised text classification leveraging experts and word embeddings.