Unsupervised Authorship Attribution on SCOTUS Opinions

1 Authorship attribution

Authorship attribution is the problem of determining who, among a set of known authors, wrote a document. It has a range of applications, including the analysis of historical texts, plagiarism detection, and the prosecution of cyber crime [1]. Authorship attribution is a subdomain of text classification with unique characteristics: stylistic features, rather than content, are relevant for classification [2].

2 The dataset

My data came from a kaggle dataset of approximately 35,000 Supreme Court of the United States (SCOTUS) opinions from the inception of the Supreme Court in 1789 until 2017 [3]. The full dataset includes dates and other information; I pruned the dataset to retain only the author and the body text of the decision. I removed very short opinions (<3000 characters), as suggested by the Kaggle dataset creator [3], because they may not be long enough to provide a good representation of the author's writing style. The fifth most prolific author wrote 815 opinions, so I took 815 opinions from each of the top five most prolific authors to create a dataset of 4075 documents where each of the five authors is equally represented.

3 Approaches in the literature

Unsupervised classification involves separating, or 'clustering', unlabelled data based on common features. Much of the literature on authorship attribution involve using labelled data to train a classifier; less work has been done on unsupervised approaches. Unsupervised authorship attribution fundamentally involves two problems: (1) how to best vectorize a piece of text, and (2) how to best cluster the document vectors.

Hachonen-Kerner and Margaliot [4] perform authorship attribution using unsupervised learning on a corpus of *Responsa*, answers to religious question written by five well-known Jewish rabbis. The nature of this corpus is similar to my dataset: all authors are highly educated in the same field, texts fall within a specific domain of discourse, and the topic of each text is unrelated to its authorship. Hachonen-Kerner and Margaliot attempt several approaches. They create document vectors three different ways: normalized term frequency (TF) of most frequent words including stop (function) words, TF of most frequent words excluding stop words, and TF of words with the highest variance variance values. They attempt two non-hierarchical clustering algorithms: K-means and Expectation Management. Their best results for 5-way clustering was 66.5%, achieved using TF of most frequent words including stop words and Expectation Management.

Mingzhe and Minghu [5] attempt both K-means and hierarchical clustering for authorship attribution on the works of five famous Chinese novelists. They try two methods of vectorizing documents: bigrams, which are two-character sequences, and term frequency of punctuation marks. Their best results for 5-way clustering were 96%, achieved by vectorizing with bigrams and clustering with hierarchical clustering using Kullback-Leibler distance and Ward linkage.

4 My clustering approach

I attempt two clustering algorithms implemented by the scikit-learn library [6]: K-means and agglomerative (bottom-up) hierarchical clustering with Euclidean distance and Ward linkage. Ward linkage means that the variance of the two clusters is what is minimized when two clusters are merged [6]. I perform clustering for 2, 3, 4, and 5 authors. Different combinations of 2, 3, and 4 authors produce different results, so for clustering with fewer than five authors I perform clustering for every possible combination.

5 Measuring classification accuracy

Because clustering is unsupervised, the clusters produced are not labelled with a correct author. I define the “correct” author of a cluster C to be the author who wrote the most documents clustered to C. Therefore multiple clusters may potentially be assigned the same author, and some authors might not be the correct author for any cluster. Having assigned a correct author for each cluster, I compute the precision, recall, and f-measure for each cluster. I also compute the total percent correct, and the improvement rate, which is equal to the total percent correct minus what percent correct is expected from random chance. I compare improvement rates to determine the “best” results.

6 Creating document vectors

I attempted many different combinations of parameters in order to determine how documents could be vectorized for best clustering results.

6.1 Tokenizing

I began by tokenizing the data using the Python natural language toolkit (NLTK) [7]. NLTK recognizes when punctuation should and should not be considered part of a word. For example, it treats the period at the end of a sentence as a distinct token from the word preceding it, but recognizes that entire acronyms like “U.S.” should be a single token. Some tokens included arabic numerals because opinions often reference other court cases or sections of the constitution or criminal code by number. These tokens were excluded.

6.2 Stop (function) words and punctuation

Stop words are function words like “and”, “if”, and “those” that do not contribute to the content of a text. Many approaches to text classification exclude stop words and punctuation. However, because authorship is based on stylistic features, I hypothesized that stop words and punctuation would be useful features for classification. I tried both including and excluding stop words, as well as using exclusively stop words. I also tried both including and excluding punctuation.

6.3 Named Entity recognition

Named Entity (NE) recognition involves identifying definite noun phrases such as certain people, organizations, dates, and places [7]. For example, in the sentence “Obama lived in the White House”, “Obama” and “White House” are NEs. Because in my corpus NEs are likely related to the specifics of a certain case rather than to authorship, I hypothesized that removing NEs would improve the results. Recognizing NEs involves constructing a syntax tree and identifying the tree nodes that correspond to NEs, which I performed using NLTK’s part of speech tagging and syntax tree chunking tools [7].

6.4 Normalized term frequency or TF-IDF

Many text classifiers use TF-IDF, which gives greater weight to rarer terms. I hypothesized that TF-IDF would be useful if certain authors use distinctive vocabulary, but that it might introduce noise if the nature of certain court cases require specialized vocabulary. I tested both TF-IDF and normalized term frequency (TF).

6.5 Vector length

Hachonen-Kerner and Margaliot [4] achieved good results using only the 1000 most frequent words. There were 68211 unique words in my dataset. I tested using the N most frequent words for many N between 500 and 68211 to determine what N achieves best results.

7 Results

I achieved the best results using normalized term frequency for the 2000 most frequent tokens including stop words and punctuation. I found that increasing N increased accuracy until a threshold around N = 2000 where increasing N began to decrease accuracy. Using TF-IDF decreased the accuracy significantly; as did excluding stop words. In fact, I found that stop words were so useful for authorship attribution that using stopwords alone for 5-way clustering achieved a 24% improvement rate. Removing or keeping Named Entities had no effect on the accuracy, likely because words that belong to Named Entities are less common and therefore many may not be included in the N most frequency words.

Hierarchical clustering achieved slightly better results than K-means. I found that the results of clustering with fewer than five authors was variable, likely because some authors have more distinct writing styles than others. The results for k < 5 in Table 1 are the averages across all combinations of k authors.

Table 1: Summary of best results using hierarchical clustering

k	Percent correct	Improvement rate
2	77.4%	27.4%
3	68.8%	35.3%
4	63.4%	38.4%

5	60.0%	40.0%

8 Running time and scalability

Hierarchical clustering is a resource-intensive algorithm; it has a time complexity of $O(n^3)$, or $O(n^2 \log n)$ using a priority queue implementation, and it requires $O(n^2)$ space. However, the nature of authorship attribution permits use of resource-intensive algorithms because the size of the datasets are limited by definition. A single author can only produce a limited amount of text, and generally authorship attribution is performed for a small set of known authors. In my case, with 5 authors, 4075 texts total, and 2897 words on average per text, the running time of vectorizing the text and performing hierarchical clustering with k=5 was approximately one minute. Tokenizing with NLTK's tokenizer, which I did separately before creating vectors and performing clustering, was the slowest process and took approximately 3 minutes for the whole dataset.

9 Future work

I found that stop word and punctuation frequency was a strong predictor of authorship, but I did not investigate stop word and punctuation sequencing. Because different authors likely structure sentences in distinctive ways, vectorizing documents according to sentence structure might be an effective approach. Modelling sentence structure could involve removing content words and creating n-grams from the sequences of stop words and punctuation. Vectors based on the most frequent n-grams could be used for clustering. Another possible avenue would be to develop a way of representing sentence or paragraph length in the document vector, as these may be stylistic differences indicative of authorship.

10 References

[1] S. Seifollahi *et al*, "Optimization Based Clustering Algorithms for Authorship Analysis of Phishing Emails," *Neural Processing Letters,* vol. 46, *(2),* pp. 411-425, 2017.
[2] I. Bozkurt, O. Gağhğlu, and E. Uyar, "Authorship Attribution: Performance of various features and classification methods," in *Proceedings of the 22nd International Symposium on Computer and Information Sciences, ISCIS 2007, Ankara, Turkey, November 7-9, 2007*. E. G. Schmidt et al, Eds. Pisataway, NJ: IEEE, 2007. pp. 158-162.
[3] G. Fiddler, *SCOTUS opinions*, Kaggle, 2019. [Online]. Available: https://www.kaggle.com/gqfiddler/scotus-opinions.
[4] Y. HaCohen-Kerner and O. Margaliot, "Authorship Attribution of Responsa using Clustering" *Cybernetics and Systems,* vol. 45, *(6),* pp. 530-545, 2014.
[5] J. Mingzhe and J. Minghu, "Text clustering on authorship attribution based on the features of punctuations usage," in *Proceedings of the 11th International Conference on Signal Processing, ICSP 2012*. DOI: 10.1109/ICoSP.2012.6492012.
[6] Scikit-learn: Machine Learning in Python, Pedregosa *et al.*, JMLR 12, pp. 2825-2830, 2011.
[7] S. Bird, E. Loper and E. Klein, *Natural Language Processing with Python*. O'Reilly Media Inc, 2009.