Authorship and date classification using syntactic tree features

Alex Cope

December 2013

Abstract

Authorship classification of documents using syntactic features has shown high levels of success; however the complexity in mining syntactic features has generally limited them to a basic feature set (POS tags, rewrite rules). S. Kim et al. [1] have proposed a novel algorithm to generated a set of syntactic features based on frequent subtree patterns of a set of syntactic trees. In this paper, I use a modified version of their algorithm to predict authorship and date of historical texts.

1 Introduction

Authorship classification and its variants (genre classification, topic classification) are nontrivial problems, even for human beings. The problem has a wide range of applications in both technical and humanities fields; authorship classification algorithms have been used to resolve texts of disputed origin such as Shakespeare's plays and the anonymous Federalist Papers [6].

Less work has been done with classifiers which predict date of publication. This is perhaps date classification is a more difficult problem than authorship publication, as it groups together documents with little in common in terms of genre and content. Any date classifier would have to take into account general trends in the evolution of the English language or fashionable topics.

Historically, the most effective approaches to authorship classification have extracted syntactic features of a document, including style markers such as function words (which are words with little lexical meaning but high structural meaning, such as *and* and *then*) and grammatical markers like part-of-speech tags and rewrite rules. Unfortunately, there have been few classifiers which extract more complex syntactic features, due to the computational complexity of algorithms involving syntactic structure.

Recently, an algorithm has been developed [1] which extracts from a set of syntactic trees a list of frequently appearing subtrees. Note that here, subtree refers to any tree fragment within a tree; the less generalized definition of a subtree being any non-root node of a tree does not apply. In this paper, I will use this novel set of syntactic tree features to test authorship and date classification on a set of historical public-domain documents.

2 Prior Works

This work was inspired initially by a CS224N project from two years ago by Andrew Tausz [5]. Tausz was interested in the same problem of date classification and used the same source (Project Gutenberg). However, he used a feature set consisting solely of lexical features like n-grams and punctuation use. I was inspired to extend his work and test date classification using more complex syntactic features.

The main source for this article is the document by S. Kim et al., 2010 [1] which presents an algorithm for mining frequent subtree patterns from a set of syntactic trees, which I will delve into with more detail in the next section. The algorithm they use is based on the CMTreeMiner algorithm [3]. The algorithm I ended up implementing was a modification of both, more efficient at the cost of poorer performance.

S. Kim et. al use their feature set to classify a set

a news articles with four different authors and a set of movie reviews with four different authors. To my knowledge, the syntactic feature set has not yet been tested on historical texts or with a date classifier.

3 Approach

3.1 Obtaining and Processing Data

All documents were retrieved from Project Gutenberg, which is an online repository of documents in the public domain. Project Gutenberg provides an RDF catalog file with information about each document. Title and author information (including, for some authors, their date of birth and date of death) for each downloaded document in the catalog file was compiled into an index. Unfortunately, the catalog file does not include the year of publication. The year of publication for a given document was extracted as follows:

- The author's Wikipedia page was retrieved, and scanned for occurrences of the document title. The most common four-digit number that followed the book title was marked as the year. (To prevent gross misestimates, any numbers not between the author's birth and death date (if those were provided) were dropped.)
- If the first step failed (a legitimate date could not be found), the book's Wikipedia page (if it existed) was retrieved, and a similar procedure was followed to extract the date.
- If the second step failed and the author's birth date / death date was provided, a heuristic was used where the date was assigned as 30 years following the author's year of birth or 30 years before his death.
- If the author's birth date / death date was not available, the document was dropped from the dataset.

To finish preprocessing, the Gutenberg-provided headers and footers (which provide legal information and use a consistent format) were cut from each document. For the syntactic tree feature set, the Stanford NLP library was used to parse the sentences of each document. Because of time concerns, only the first 1,000 sentences of each document was parsed. In total, 1,000 documents were retrieved from Project Gutenberg and 682 were used in the dataset after removing duplicate files and files without retrievable year of publication information, representing 360 MB of data.

3.2 Baseline Features

For my baseline feature set I extracted unigram (word) features from each document in the dataset. To prevent unnecessarily large feature vectors, words are only used as features if they appear at least times in the training data set. The value of each feature per document is the number of times that word appears in the document; this results in a sparse and high-dimensional feature vector. Only the first 10,000 words were scanned for each document.

3.3 Mining Closed Subtree Patterns

The number of frequent subtrees in a syntactic tree grow exponentially with the tree size. Thus, it would be computationally inefficient to iterate through every subtree in a syntactic tree set and test each individually for frequency. The authors of CMTreeMiner have proposed an algorithm which recursively mines for frequent subtrees by extending previously found frequent subtrees in a depth-first search manner.

To understand the steps of the algorithm it is necessary to first present a few definitions. We say that a tree t is a subtree of s if all the vertices and edges of t are subsets of s. The ***occurrence*** of t in s is the number of distinct subtrees t that are present in s. s ***supports*** pattern t if the occurrence of t in s is at least 1, and the *support* of a pattern t is the number of trees in a dataset that support t, i.e. $supp(t) = \sum_{s \in D}(s \text{ supports } t)$. A tree t is ***frequent*** if its support is greater than a minimum support number *minsup* given by the user. When mining subtrees, we seek to find all frequent subtrees in a database. One helpful property of frequent trees is

that all supertrees of an infrequent tree are infrequent and all subtrees of a frequent tree are frequent.

We define the ***blanket*** of t B_t as the set of all supertrees of t with one more vertex than t. A frequent tree t is ***closed*** iff for every $t' \in B_t$, $supp(t') < supp(t)$. In other words, a closed frequent subtree pattern is one in which none of its supertrees occur in the same number of trees in the dataset. By mining only closed patterns, we remove redundant patterns from the generation process, speeding up the algorithm exponentially and drastically decreasing the feature space with zero loss of information. We say that $t' \in B_t$ and t are ***occurrence-matched*** if for each occurrence of t in a set of syntactic trees there is a corresponding occurrence of t', and t' and t are ***support-matched*** if each tree s in a set of syntactic trees that supports t also supports t'.

Finally, we can divide the blanket of t into the ***right-blanket*** B_t^r, which consists of all $t' \in B_t$ where the extra vertex is the rightmost vertex of t', and the ***left-blanket*** $B_t^l = B_t - B_t^r$.

With all of these definitions in mind, the closed pattern mining algorithm is given below. For further discussion on the theorems behind the algorithm, refer to Y. Chi et al. 2004 [3].

Algorithm 1 ClosedPatternMiner $(D, minsup)$

```
CL ← 0
C ← frequent 1-trees ∈ D
CM-Grow(C, CL, minsup)
return CL
```

3.4 SVM Classifier

I trained both feature sets using support vector machine (SVM) classifier, using the popular free C-based implementation SVMlight. Because both classification problems are instances of multiclass classification (distinguishing between m authors and n date periods), I used the one-vs-all approach, which has been shown to be a computationally efficient and accurate approach to SVM classification [4]. In this approach, n different binary classifiers are trained on the same train data set, each one trained to distinguish one class from all other $n-1$ classes. When classifying a new example, it is run on all n classifiers, and it is assigned to the classifier which outputs the maximum value. Each document in the dataset was randomly placed into either the training set (with 70% probability) or the test set. After testing with multiple kernels, I found that the radial basis function (Gaussian) kernel ($\exp(-\gamma\|a-b\|^2)$) was by far the highest performant.

Algorithm 2 CM-Grow $(C, CL, minsup)$

```
for i ← 1, ..., |C| do
    E ← 0
    if there exists c' ∈ B_c^l that is occurrence-matched with c_i then
        continue
    end if
    if there is no c' ∈ B_c that is support-matched with c_i then
        CL ← CL ∪ c_i
    end if
    for e ∈ B_c^r do
        if supp(e) > minsup then
            E ← E ∪ e
        end if
    end for
end for
return
```

4 Results

4.1 Performance of Feature Sets

In both the authorship classification test and the date classification test, the unigram model performed better than the syntactic tree pattern model. Tables 1 and 2 illustrate the results of a authorship classifier trained on subtree features and unigram features, respectively. The five authors with the most documents in the dataset (Robert Louis Stevenson, L. Frank Baum, Henry James, Edgar Rice Burroughs, and Charles Dickens) were compared for the test. The number next to each author's name in the table represents the number of documents in the test set positively labeled as that author. The subtree clas-

Author	P	R	F1
Robert L. Stevenson (N = 4)	0.75	0.50	0.60
L. Frank Baum (N = 7)	1.00	1.00	1.00
Henry James (N = 4)	1.00	1.00	1.00
Edgar R. Burroughs (N = 8)	0.80	1.00	0.89
Charles Dickens (N = 4)	0.83	0.83	0.83
Average	0.88	0.87	0.86

Table 1: Authorship classifier trained on subtree patterns.

Author	P	R	F1
Robert L. Stevenson (N = 4)	0.15	1.00	0.27
L. Frank Baum (N = 7)	0.00	0.00	0.00
Henry James (N = 4)	0.00	0.00	0.00
Edgar R. Burroughs (N = 8)	1.00	0.12	0.22
Charles Dickens (N = 4)	0.00	0.00	0.00
Average	0.23	0.23	0.10

Table 2: Authorship classifier trained on unigram features.

sifier drastically outperformed the unigram classifier, with an F1 score of 0.86 vs. 0.10.

Tables 3 and 4 show the results of a date classifier trained on subtree features and unigram features, respectively. For this test, all 682 documents from the dataset were used. Dates were separated into four buckets, as indicated in the tables. Again, the subtree classifier performed better than the unigram classifier, with an F1 score of 0.36 vs. 0.17.

4.2 Discriminative Subpatterns

It is informative to see which subtree patterns are most discriminative. The Fisher score of a pattern

Period	P	R	F1
<= 18th century (N = 18)	0.80	0.14	0.24
Early 19th century (N = 29)	0.42	0.69	0.52
Late 19th century (N = 75)	0.54	0.46	0.49
20th century (N = 81)	0.50	0.11	0.18
Average	0.56	0.35	0.36

Table 3: Date classifier trained on subtree patterns.

Period	P	R	F1
<= 18th century (N = 18)	1.00	0.04	0.07
Early 19th century (N = 29)	0.40	1.00	0.57
Late 19th century (N = 75)	1.00	0.01	0.02
20th century (N = 81)	0.00	0.00	0.00
Average	0.60	026	0.17

Table 4: Date classifier trained on unigram features.

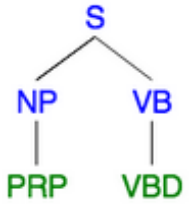

Figure 1: An example subtree pattern with a high Fisher score. (0.127)

is a good metric to evaluate its discriminative power. It is defined as

$$Fr = \frac{\sum_{i=1}^{c} n_i(\mu_i - \mu)^2}{\sum_{i=1}^{c} n_i \sigma^2}$$

where n_i is the number of documents in class i, μ_i is the average frequency of a feature in class i, σ_i is the standard deviation of a feature in class i, and μ is the average frequency over all classes. A large Fisher score indicates that a pattern has similar values to documents in its own class and different values from documents in other classes.

For example, the subtree pattern in Figure 1 has a Fisher score of 0.127 (the 13th highest of all patterns) and is present in 0.11% of the 18th century documents, 0.10% of the early 19th century documents, 0.18% of the late 19th century documents, and 0.21% of the 20th century documents. For comparison, the words with the highest Fisher scores in the unigram model include function words like *that* and *which*, and more obviously indicative words like *thou*, *God*, and *thy*.

5 Discussion

In both cases the subtree feature set outperformed the baseline unigram feature set. The experiment thus confirms that syntactic structure is a strong indicator of authorship. However, the date classification average F1 score is still disappointingly low. In the following sections I discuss some potential reasons for the low score and discuss ways to potentially mitigate these issues in future experiments.

5.1 Dimension Reduction

SVMs are subject to overfitting if the dimension size of a vector space is too large. Although I calculated the Fisher score of each pattern, this was primarily for interest and manual examination of discriminative patterns. Using the Fisher score of patterns to further cull the list of discriminative patterns would reduce the dimensionality of the final feature vector and result in a more efficient and potentially more accurate feature set. Overfitting is a likely issue here, as when I tested the dataset on the training set the resultant F1 score was 0.86, which is much higher than the F1 score of the test set (0.36).

Furthermore, a different classifier may have been more appropriate for these classification problems. With their independence assumptions, Naive Bayes classifiers are less prone to overfitting than SVMs and they have been shown in some cases to perform better than SVM in authorship and other classification tasks.

5.2 Minimal Data

For the date classification dataset, only the first 100 sentences of each document was parsed. This was due to an memory overflow error in the Java code when trying to import the full 1.16 GB file of parses of the first 1,000 sentences that I could not resolve in time. This may not be enough to get a good, unbiased representation of a given class. Parsing the full document or parsing more documents may help improve SVM accuracy.

Furthermore, the closed patterns were generated from a much smaller set of trees; only about ten per document were used because of the enormous computational complexity of generating subtree patterns. My justification was that even with such a limited data set, frequent sub patterns of support 3 could still be found and generated; however, this may have limited my subtree patterns to smaller, more generally common, less discriminative fragments.

5.3 Algorithmic Shortcuts

Due to time constraints, I implemented many shortcuts in my feature extraction algorithm to help reduce computation time. For instance, after generating 500 closed patterns I stopped the pattern miner early. Because of the exponential nature of the algorithm, it is possible that thousands of closed patterns could have been generated; furthermore, because the algorithm iterates successively through each frequent 1-tree, the subset of closed patterns I generated was not a good sample, as it only includes patterns with roots from the first few members of the list of frequent 1-trees.

5.4 Feature Combination

The feature set of subtree patterns uses those patterns exclusively. A feature set that combines subtree features with other well-performing features (eg. unigrams, function words, sentence length) might perform better than any single feature set.

5.5 Fundamental Issues

One fundamental question remains: is sentence structure a good indicator of the date of publication of an article? Has sentence structure evolved over the course of the English language in a predictable and classifiable way? Certainly, certain words were more used in the past (eg. *thou* and *thy*) than they are today; it is less obvious that certain subpatterns of syntactic trees have fallen in or out of style over the years. Further experiments that optimize the classifier I've presented here should be conducted to determine the viability of sentence structure as a marker for date of publication.

As a side note, it's interesting that the precision of the 18th century period bucket was so much relative to the rest of the buckets. This seems to demonstrate that there are at least some documents from that period with significantly difference sentence structures than all 19th century and beyond documents. Historically, this makes some sense; the English language, in both spelling and grammar, was not properly standardized until the late 18th century, so it would follow that sentence structure before that period would be a stronger indicator of date.

References

[1] Sangkyun Kim, Hyungsul Kim, Tim Weninger, and Jiawei Han, *Authorship classification, a syntactic tree mining approach* Proceedings of the ACM SIGKDD Workshop on Useful Patterns (New York, NY, USA). UP'10, ACM, 2010, pp. 65-73.

[2] Joachim Dietrich, Jrg Kindermann, Edda Leopold, and Gerhard Paass, *Authorship Attribution with Support Vector Machines* Applied Intelligence 19, pp. 109-123, 2003.

[3] Yun Chi, Yirong Yang, Yi Xia, and Richard R. Muntz, *CMTreeMiner: Mining Both Closed and Maximal Frequent Subtrees*, Advances in Knowledge Discovery and Data Mining pp. 63-73, 2004.

[4] Ryan Rifkin and Aldebaro Klautau, *In Defense of One-vs-All Classification* Journal of Machine Learning Research 5, pp.101-141, 2014.

[5] Andrew Tausz, *Predicting the Date of Authorship of Historical Texts*, CS224N project, 2011.

[6] D. I. Holmes and R. S. Forsyth, *The Federalist Revisited: New Directions in Authorship Attribution* Lit Linguist Computing 2, pp. 111-127, 1995.