Authorship Classification: A Syntactic Tree Mining Approach *

Sangkyum Kim, Hyungsul Kim, Tim Weninger, Jiawei Han
University of Illinois at Urbana-Champaign
{kim71, hkim21, weninge1, hanj}@illinois.edu

ABSTRACT

In the past, there have been dozens of studies on automatic authorship classification, and many of these studies concluded that the writing style is one of the best indicators of original authorship. From among the hundreds of features which were developed, *syntactic features* were best able to reflect an author's writing style. However, due to the high computational complexity of extracting and computing syntactic features, only simple variations of basic syntactic features of function words and *part-of-speech* tags were considered. In this paper, we propose a novel approach to mining discriminative k-embedded-edge subtree patterns from a given set of syntactic trees that reduces the computational burden of using complex syntactic structures as a feature set. This method is shown to increase the classification accuracy. We also design a new kernel based on these features. Comprehensive experiments on real datasets of news articles and movie reviews demonstrate that our approach is reliable and more accurate than previous studies.

Categories and Subject Descriptors

I.2.7 [**Artificial Intelligence**]: Natural Language Processing—*Text Analysis*; H.3.3 [**Information Search and Retrieval**]: Clustering; H.2.8 [**Database Applications**]: Data Mining

General Terms

Algorithms, Pattern

*This research is part of the Blue Waters sustained-petascale computing project, which is supported by the National Science Foundation (award number OCI 07-25070) and the state of Illinois. Blue Waters is a joint effort of the University of Illinois at Urbana-Champaign, its National Center for Supercomputing Applications, IBM, and the Great Lakes Consortium for Petascale Computation. This work was also sponsored in part by the National Science Foundation (under grants IIS-09-05215, CCF-0905014, and CNS-0931975) and an NDSEG Fellowship award. Any opinions, findings, and conclusions or recommendations expressed here are those of the authors and do not necessarily reflect the views of the funding agencies.

UP'10, July 25th, 2010, Washington, DC, USA.

Keywords

Authorship Classification, Text Mining, Text Categorization, Discriminative Pattern, Closed Pattern

1. INTRODUCTION

In computational linguistics and text mining domains, there have been three classical classification problems: topic classification, genre classification, and authorship classification. Among those three problems, the most difficult one is encountered when we try to classify documents in terms of their authorship (known as *authorship classification*, authorship attribution and/or authorship discrimination). This problem can be thought of as classifying documents based on the writing styles of the authors. It is a nontrivial problem even for the human beings: while a human can easily identify the topic and genre of a document, identifying its authorship is harder. Even worse, if the documents are from the same topic and genre, the task becomes much harder.

In the era of excessive electronic texts, authorship classification has been more and more important with a wide variety of applications. Besides the early works of analyzing disputed plays of *Shakespeare*(1887) [20] or anonymous documents of *The Federalist Papers*(1964) [23], it could also be used to identify authors of short 'for sale' messages in a newsgroup [37] and even for forensic investigations by identifying authorship of e-mail messages [2]. Detecting plagiarism or copyright infringement of unauthorized reuse of source code by establishing a profile of an author's style is another important application of authorship classification [6].

Existing approaches of authorship classification use various methods to extract effective features, most commonly style markers such as function words [11, 35, 1, 13] and grammatical elements such as part of speech (*POS*) tags [3, 12, 36]. Function words are the most common words that have little semantic content of their own but usually indicate a grammatical relationship or generic property. The success of using function words and *POS* tags as features for authorship classification indicates the usefulness of syntactic information.

Unfortunately, research on more complex syntactic structures has not yet been flourished because of the lack of a reliable, automatic tool which retrieves syntactic structures, and because of the high computational cost associated with syntactic structure-based algorithms. Instead, several rather simple syntactic structures, such as rewrite rules [3, 12] and n-grams of *POS* tags [11, 12, 15, 14] were discussed.

Lately, several advanced techniques were developed which greatly improved the performance of *Natural Language Pro-*

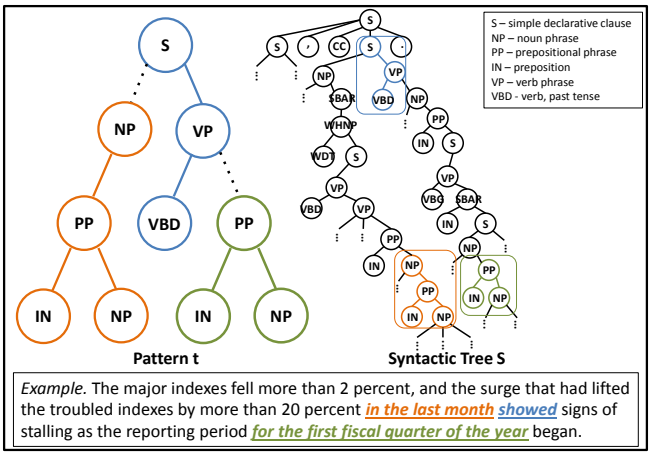

Figure 1: A 2-*ee* subtree pattern t is mined from two *NY Times* journalists Jack Healy and Eric Dash who worked in the same business department. On average, 21.2% of Jack's sentences contained t while only 7.2% of Eric's sentences contained t.

cessing(NLP) tools [18] enabling reliable, highly accurate sentence parsing into a syntactic tree of POS tags. Recently, emerging research of question answering (QA) systems [22, 4, 30] adapted these advanced preprocessing techniques to develop a tree kernel function that computed a matching score of two syntactic trees in order to retrieve similar questions or to classify questions in the QA system. But those approaches have several problems to be applied to authorship classification problem for the following reasons: (i) Even though their tree kernel utilizes more complex features than earlier works of rewrite rules and n-grams of POS tags, those features were in restricted forms of subtrees of a syntactic tree. There is a need to design a feature set that can capture syntactic information of a longer and more complicated sentence structure than simple question formats. (ii) The number of features becomes explosive once we consider all possible subtrees (even with some restrictions), and it leads to a burden on computational cost, however efficient a kernel computation is. (iii) Existing tree kernels work only for a data set of trees, not a data set of sets of trees. A question can be transformed into a syntactic tree, but a document which consists of a set of sentences becomes a set of syntactic trees.

In this paper, we propose a novel syntactic feature set of tree fragments allowing at most k-embedded edges (in short, k-*ee* subtree). Compared with previous feature sets that consists of distinct subtree components, our new feature set captures the relationship between $k+1$ subtree components of a syntactic tree, which leads to a better representation of a data set of long and complex sentences. To reduce the number of features, we only mine a set of discriminative and frequent k-*ee* subtrees, which results in higher accuracy by avoiding overfitting to the training data and by not generating non-discriminative features that often deteriorate the performance. For the classification, we introduce a new tree kernel by defining a proper value for each corresponding feature to be well-defined and effective on a data set of sets of trees.

Figure 1 gives an example of a k-*ee* subtree pattern t for $k = 2$. Pattern t is composed of three smaller induced subtrees, which are connected by two embedded edges (S,NP) and (VP,PP). The differences of pattern distributions between two authors suggest that a set of k-*ee* subtree patterns can be utilized as a good feature set for authorship classification.

Our framework can also be considered as a tree-kernel method, but it is different from previous tree-kernel approaches of a QA system in the following ways: First, our objects to be classified are in a more general form. Previous tree-kernel methods work for questions where each question becomes one syntactic tree, while our approach are based on documents where each document is a set of syntactic trees. Since previous tree kernels work only between two syntactic trees not between two set of syntactic trees, it cannot be directly applied to the authorship classification problem. Second, we use a more general feature set of subtree patterns allowing k-embedded edges, which works well to represent long and complex syntactic structures of a sentence. Third, a tree-kernel method essentially matches two trees without looking at the entire dataset. That is, it counts the common number of subtree patterns of two syntactic trees. But, our approach can get an overview of different classes of training data to select the discriminative patterns as features.

We adapt the framework of discriminative frequent pattern mining which showed good results for various problem settings in unstructured and semi-structured data mining such as mining discriminative frequent itemset, sequence, and graph patterns to classify UCI datasets, software behaviors, and chemical compound data, respectively [8, 19, 31].

While other syntactic features utilize the *bag-of-words* model to represent a document – which assigns the number of occurrences of a feature to its value – k-*ee* subtree patterns cannot adapt the same way due to the overlapped occurrences. Since we consider all subtrees and even allow k-embedded edges, a huge number of occurrences might overlap each other which would lead to an exaggeration of a feature value. At the other end, binary features will lose most of their occurrence information which results in either 0 or 1. Therefore, we design a new way to assign proper values for k-*ee* subtree features in between two extreme ends.

To validate the utility of our new feature set to others, for fair comparisons, we apply the same classification algorithm (SVM) to various feature sets over several real datasets. Experimental results demonstrate the effectiveness of our newly proposed feature set of k-*ee* subtree patterns over the well-known existing feature sets.

In summary, the contributions of this paper are as follows:

- We propose a new feature set of k-*ee* subtree patterns for authorship classification.
- We develop an algorithm to mine discriminative k-*ee* subtree patterns.
- We propose a new document representation based on our new feature set of k-*ee* subtree patterns that is proper for data of sets of trees.
- Through experiments on various datasets, we demonstrate the utility of our proposed framework to provide an effective solution for the authorship classification problem.

The rest of the paper is organized as follows. Section 2 presents an overview of the related works. In Section 3, we introduce various preliminary concepts, define our new feature *k-ee* subtree pattern, and describe our *k-ee* subtree pattern-based authorship classification framework. Section 4 presents a closed and frequent *k-ee* subtree mining algorithm with several pruning techniques. In Section 5, we explain discriminative pattern mining with a sequential coverage approach. We report our experimental results in Section 6, followed by conclusions and future work in Section 7.

2. RELATED WORKS

There are two main steps involved in any authorship classification algorithms: feature extraction step and classification step based on extracted features. For the feature extraction step, since the earliest works that used a small number of common words such as '*and*', '*to*' as a feature set, nearly 1,000 different features have been studied including sentence length, chi-square score, lexical richness [25, 17], vocabulary richness [10], function words [1], word n-grams [26], character n-grams [14], and rewrite rules [3] with lots of controversy on their effectiveness. Even though there was an issue of fair comparison between feature sets because previous works conducted experiments based on their own data sets with different classification methods [35, 27], function words and rewrite rules were considered to show reliable results. In [27], comprehensive survey on different feature sets were presented.

For the classification step, even though lots of new features were explored for authorship classification, most of the classification algorithms were simply adapted from well-known classification algorithms in other domains such as *PCA* [16], *k*-nearest neighbor, decision tree, bayesian networks [35], language model [36], and *SVM* [11, 12, 36, 15]. The ones that showed good performance in other fields like language model method and *SVM* also showed high accuracy for authorship classification. For this reason, usually *SVM* has been used to compare the effectiveness of feature sets [12, 15], so in this paper we also use *SVM* for fair comparison between our new feature set and previous feature sets.

Our proposed feature set of *k-ee* can be considered as a variation of tree patterns. In data mining domain, there have been several studies on tree pattern mining [33, 9, 28]. *TreeMiner* [33] is one of the pioneer of mining frequent tree patterns. *CMTreeMiner* [9] mined closed and maximal frequent tree patterns together.

For tree classification, rule-based classifiers (*XRules* [34], *CTC* [38]) and a decision tree based classier (*Tree*^{2} [5]) were proposed. But none of them could be applied to classify sets of trees as documents.

3. PRELIMINARIES

Traditional authorship attribution approaches adopted function words, *POS* tags, and rewrite rules as a feature set to build a classification model. Even though they achieved good accuracy, there still existed room to find a more meaningful feature set to improve the performance. In this section, we describe rewrite rules which are somewhat complex syntactic structures that hold more syntactic information than the other two feature sets. Secondly, we define our new feature set of *k-ee* subtree patterns.

3.1 Rewrite Rule

In [3], rewrite rules were considered to be building blocks of a syntactic tree, just as words are building blocks of a sentence. Here, a *syntactic tree* is a rooted and ordered tree which is labeled with *POS* tags that represents the syntactic structure of a sentence. Its interior nodes are labeled by nonterminals of the grammar, and the leaf nodes are labeled by terminals.

Compared to previous approaches that utilized function words and *POS* tags, rewrite rules can hold functional structure information of the sentence. In linguistics, a rewrite rule is in the form of "$X \rightarrow Y$" where X is a syntactic category label and Y is a sequence of such labels such that X can be replaced by Y in generating the constituent structure of a sentence. For example, "$NP \rightarrow DT{+}JJ{+}JJ{+}NN$" means that a noun phrase (NP) consists of a determiner (DT) followed by two adjectives (JJ) and a noun (NN).

There is a limit when using rewrite rules as features of a classification model. First, because of the restriction that the entire rule cannot be broken into smaller parts, no similarity between rules are considered. A large number of slightly different rules are all counted as independent features. For instance, a rewrite rule "$NP \rightarrow DT{+}JJ{+}NN$", missing one JJ from the above example, becomes a separate rewrite rule. Second, since a rewrite rule is a two-level tree structure, it is not enough to hold most of the syntactic structure information of a sentence. For example, the relationships between rewrite rules are missing, which can hold more refined syntactic information. For these reasons, we developed a new feature set of *k-ee* tree patterns that are flexible and complex enough to represent the syntactic structure information of a sentence.

3.2 k-Embedded-Edge Subtree Pattern

To overcome the drawbacks of the feature sets used in previous approaches, we extended the definition of the rewrite rule to form a new feature set. Based on the analysis of the rewrite rule, a new feature should be a multi-level tree structure to hold the novel information of the syntactic structure of a sentence. Moreover, it should be allowed to contain only a part of a rewrite rule. Induced subtree patterns of a syntactic tree were one of the candidate feature set which satisfied both conditions. But, our pilot experiments showed that a small number of combinations of those induced subtree patterns could achieve even higher accuracy, which motivated us to define *k-ee* subtree patterns for our new feature set as follows.

Definition 1. We define a tree t to be an ***induced subtree*** of a tree s if there exists an identity mapping from t to s preserving all parent-child relationships between the nodes of t. We define an edge e of a tree s to be ***embedded*** *iff* e is a pair of two nodes of s with an ancestor-descendant (not parent-child) relationship. We define a ***k-embedded-edge subtree*** (***k-ee subtree***) t of a tree s to be a set of induced subtrees of s that can be connected by at most k embedded edges.

Since we allow a *k-ee* subtree pattern to be not only a two-level but also a multi-level subtree structure, the number of *k-ee* subtree patterns would be exponential on the number of trees and their sizes. We define a minimum support to ensure we only mine general common patterns that will be applicable to test data thus avoiding overfitting.

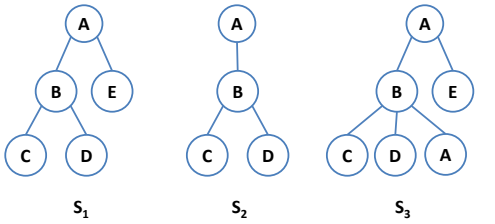

Figure 2: A toy example of a database $\mathcal{D}$ with three syntactic trees

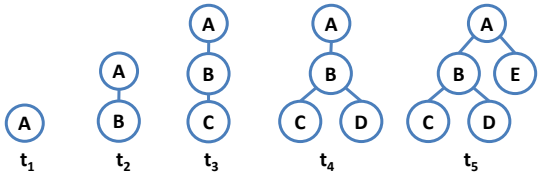

Figure 3: Examples of frequent k-*ee* patterns in $\mathcal{D}$ when $k = 0$ and $\alpha = 2$

Definition 2. We define the ***support*** of a k-*ee* subtree pattern t (denoted by $sup(t)$) to be the total number of syntactic trees of sentences in training data that contains t. We say t is ***frequent*** *iff* $sup(t) \geq \alpha$ for a user-specified minimum support threshold α.

As common words or function words were studied as features for authorship classification in previous works, frequent patterns share the philosophy that more general features are preferred to discriminate the writing styles of the authors.

Figure 2 shows a toy database $\mathcal{D}$ of three syntactic trees. Given minimum support threshold 2, all five 0-*ee* subtree patterns presented in Figure 3 become frequent. For example, patterns t_1, t_2, t_3, and t_4 appears in all three syntactic trees, so their supports are all 3. Pattern t_5 only appears in S_1 and S_3, so its support becomes 2.

Even though we only use frequent k-*ee* subtree patterns as a feature set for a classification model, the potential number of patterns can still become a bottleneck. To address this problem, we introduce the concept of a closed pattern in order to prevent generating redundant patterns; in this way we can summarize frequent patterns into a smaller set of closed patterns without any loss of information.

Definition 3. We define a k-*ee* subtree pattern t to be ***closed*** if there exists no tree pattern t' that contains t with $sup(t') = sup(t)$.

For example, two 0-*ee* subtree patterns t_4 and t_5 in Figure 3 are closed in the toy database $\mathcal{D}$ since their superpatterns have smaller support. Patterns t_1, t_2, and t_3 are not closed since they have the same support with their superpattern t_4.

We will explain how to mine closed k-*ee* subtree patterns efficiently utilizing several pruning techniques in Section 4. In this way, using closed and frequent k-*ee* subtree patterns as a new feature set not only reduces the size of the feature set but also makes our authorship classification framework more scalable.

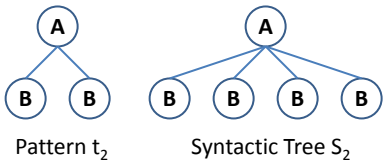

Figure 4: An example of overcounting of overlapped k-*ee* subtree pattern occurrences

3.3 Frequency Measure of k-ee Subtree

The frequency of a pattern within a document (or a set of syntactic trees) is quite important in the sense that it can be a good measure to discriminate the writing styles of different authors. Previously well-known features such as function words, *POS* tags, and rewrite rules adapted *bag-of-words* approach that used the number of their occurrences in a document as their frequency measure. However, the k-*ee* subtree patterns cannot simply adapt the same frequency measure because it generates many overlapped occurrences, which would lead to an exaggerated frequency measure. Overlapped patterns appear because we consider all kinds of subtrees allowing several embedded edges. Figure 4 is an illustration of this overcounting problem. The syntactic tree S has only one A and four Bs, but the number of occurrences of pattern t becomes 6. More generally, if A has n Bs as its children in S, then the occurrence count of pattern t becomes $O(n^2)$. Since we allow k embedded edges for a k-*ee* subtree pattern, this overcounting problem will be even more amplified.

Our observation that a document is parsed into a set of syntactic trees (of sentences) gave us an insight to define the frequency measure of a k-*ee* subtree pattern in a different way by counting the number of syntactic trees of a document that contain the pattern.

Definition 4. We define the ***frequency*** of a k-*ee* subtree pattern t in a document d (denoted by $freq(t, d)$) to be the fraction of the number of syntactic trees of sentences in d that contains t.

For example, if a document d is composed of S_1, S_2, and S_3 in Figure 2, then the frequencies of patterns t_1, t_2, t_3, and t_4 (in Figure 3) in d become all 1 while the frequency of a pattern t_5 in d becomes 2/3.

Note that our feature value is a normalized score in the sense that we only consider the fraction of the number of sentences in a document. In this way, we can remove the effect of different document lengths.

3.4 k-ee Subtree-based Authorship Classification

We propose a k-*ee*-subtree pattern-based authorship classification framework with the following four steps: (1) Convert each document into a set of syntactic trees. As mentioned earlier, several high-quality parsing tools have been developed recently. (2) Mine frequent k-*ee* subtree patterns of the syntactic trees from the training data. There are several reasons we use only frequent patterns. First, we do not assume the parser works perfectly with no error, but we do assume it works with a reasonable accuracy. A small rate of error might produce strange patterns with low support. Therefore, if we only use frequent patterns, we can reduce

the influence of parsing errors. Second, using patterns with low support as features may cause overfitting and subsequently harm the classification accuracy. Statistically, using frequent patterns of training data as features for the classification model generalizes well to the test data, since frequent patterns of training data have a higher chance to also appear in test data. (3) Select discriminative patterns from the frequent k-*ee* subtree patterns. Depending on the user specified minimum support threshold, we might get a large number of frequent patterns which may again cause overfitting. Therefore, we carefully choose only a small number of non-redundant and highly discriminative patterns as the features for the classification model. (4) Construct the classification model with the discriminative patterns and training data.

4. K-EE SUBTREE PATTERN MINING

In the previous section, we explained the reasons to use k-*ee* subtree patterns as a new feature set of authorship classification. These patterns hold more profound syntactic information (than other features including rewrite rules) and are flexible enough to consider partial matching of the syntactic trees. Even though the k-*ee* subtree patterns are confined to be frequent and closed, the number of patterns can still be very large. Therefore, the next task is to mine these patterns efficiently.

In this section, we introduce a k-*ee* subtree pattern mining method that (i) finds the frequent and closed patterns efficiently and (ii) captures their frequencies in each document. We do not generate candidate k-*ee* subtree patterns and check for frequent and closed attributes. Instead, we find a frequent k-*ee* subtree pattern, and extend it by adding a node (that is guaranteed to be frequent) in a depth-first search manner. Depth-first search pattern expansion enables several pruning techniques for closed and frequent pattern mining. We first introduce how to efficiently find a frequent node for pattern extension, and then explain the pruning techniques for closed pattern mining.

4.1 Pattern-Growth Approach

Previous apriori-based approaches for pattern mining generated a huge number of patterns and re-scanned the entire database each time the size of candidate patterns were increased to verify whether they were frequent. Recently, several studies were conducted on the pattern-growth approach by using the projected database in sequence and tree pattern mining [24, 39]. In this study, we adapt these pattern-growth techniques for frequent k-*ee* subtree pattern mining.

Instead of the apriori-based expensive candidate generation and test framework, we follow the following steps for pattern-growth approach. First, find a size 1 frequent k-*ee* subtree t in the training dataset $\mathcal{D}$. Second, project the postfix of each occurrence of t in the syntactic trees of $\mathcal{D}$ into a new database $\mathcal{D}_t$. Here, we say a ***postfix*** of an occurrence of t in a syntactic tree s to be the forest of the nodes of s appearing after the occurrence of t in a pre-order scan of s. Third, find a frequent node v in $\mathcal{D}_t$ that can be attached to the rightmost path of t that forms a k-*ee* subtree pattern. Once v is frequent in $\mathcal{D}_t$, it ensures that the extended pattern is also frequent, so we do not need to scan the whole database $\mathcal{D}$ again. The reason we only search for a frequent node that can be attached to the rightmost path of t is to avoid generating duplicated patterns in the mining process. Note that, in this study, we consider a node v attached to t by an induced edge forms a different pattern from the one attached by an embedded edge because of the k-embedded edge restriction. So we consider each case separately. Fourth, recursively go back to second step with the extended pattern for every frequent node we found. Fifth, recursively go back to the second step to expand all the other size-1 frequent k-*ee* subtrees.

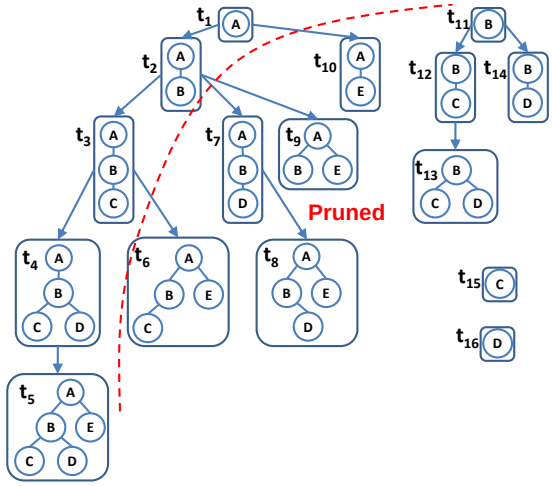

Figure 5: Pattern growth of k-*ee* subtree patterns using pruning with minimum support 2 when k=0

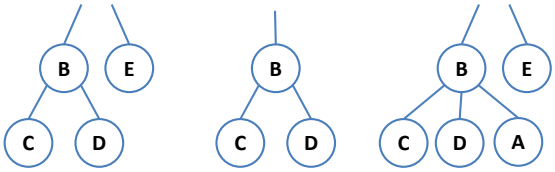

Figure 6: Projected database of t_1

4.2 Pruning Methods

Figure 5 shows an example of the pattern-growth approach to mine 0-*ee* subtree patterns of a database of three syntactic trees described in Figure 2 when the minimum support threshold is 2. Each pattern is indexed in pattern-generation order. We first search for size-1 frequent patterns, which are t_1, t_{11}, t_{15}, and t_{16} in this case. We choose t_1 as a starting point, and find frequent nodes that can be attached to t_1 from its projected database described in Figure 6. We find nodes B and E are frequent, and we extend t_1 to t_2 by adding a node B. Once all frequent 0-*ee* subtree patterns that extends t_2 are all mined, then we extend t_1 to t_{10} by adding a node E. Similar procedures are recursively performed until we mine all frequent patterns.

In our pattern-growth approach, the projected database of a pattern t keeps shrinking as the mining process moves on and t becomes a bigger superpattern. Note that we do not physically create projected databases. In fact, instead of keeping physical copy of postfix data, we use a *pseudo-projection* that only stores a pointer to the syntactic tree and the offset of each node of a pattern occurrence in the syntactic tree to save memory and make the procedure more efficient.

Algorithm 1: Procedure *ClosedMine* to mine *k*-ee closed subtree patterns

input : Tree data set $\mathcal{D}$, minimum support θ
output: Closed k-ee subtree patterns $\mathcal{C}$

```
1 foreach frequent vertex t ∈ D do
2 |  ClosedMine_Sub(t,D_t,θ);
3 end
```

Algorithm 2: Subprocedure ClosedMine_Sub used for *ClosedMine*

```
1  if t satisfies BEP condition then return;
2  if no t' ∈ B_t sentence-matches with t then
3  |  C ← C ∪ {t};
4  end
5  foreach t' ∈ B_{t_right} do     /* bottom up to enable FEP
   pruning */
6  |  if t' satisfies FEP condition then break;
7  |  if sup(t') ≥ θ then
8  |  |  ClosedMine_Sub(t',D_{t'},θ);
9  |  end
10 end
```

After we perform the pattern-growth method to mine all frequent *k-ee* subtree patterns, we can remove the patterns which are not closed. Instead of the inefficient two-step approach, we can integrate several well-known pruning techniques for semi-structured data mining [32, 29, 9] into the pattern-growth method to output only closed patterns. The common intuition of the pruning methods is that we only need to check immediate supertrees of a tree pattern t not the whole supertree for the closure checking. In this paper, we adapt the *blanket* convention of [9] to describe pruning techniques for closed *k-ee* subtree pattern mining as follows.

Definition 5. Define the ***blanket*** of a *k-ee* subtree pattern t (denoted by B_t) by the set of supertrees of t that has one more node than t. For a pattern $t' \in B_t$, we denote $t' \backslash t$ to be an additional node v of t' that is not in t. Here, $t' \backslash t$ represents not only the vertex label of v, but also its position and the type of edge connection (either induced or embedded) between t and v. We define the ***right-blanket*** of t (denoted by B_t^r) as a subset of B_t where $t' \in B_{t\_right}$ *iff* $t' \backslash t$ is the rightmost vertex of t'. We define the ***left-blanket*** of t (denoted by B_t^l) by $B_t^l = B_t - B_t^r$. For $t' \in B_t$, we define t' and t to be ***occurrence-matched*** if, for each occurrence of t in a database, there is at least one corresponding occurrence of t'. We define t' and t to be ***sentence-matched*** if for any syntactic tree s of a sentence in $\mathcal{D}$ that contains t it also contains t'.

For example, in Figure 5, pattern t_4 is in the blanket of t_7 since t_4 is a superpattern of t_7 by one more node C. And also, pattern t_4 is in both $B_{t_7}^l$ and $B_{t_3}^r$. Since $t_4 \in B_{t_7}$ and each occurrence of t_7 is contained in an occurrence of t_4, we say t_4 and t_7 are occurrence-matched.

The following two pruning techniques are based on occurrence-level matching. *Backward Extension Pruning*(BEP) checks the occurrence matching of the current mining tree pattern t with previously mined supertrees of t, and *Forward Extension Pruning*(FEP) checks the occurrence matching of t with the supertrees of t that will be mined later.

PROPOSITION 1. **(Backward Extension Pruning)** *For a k-ee subtree pattern t, if there exists a supertree $t' \in B_t^l$ such that t and t' are occurrence-matched, then neither t nor any supertrees of t as extensions of any node of its rightmost path can be closed.*

For example, pattern t_7 and its descendants in Figure 5 are pruned since t_4 is in the left blanket of t_7, and t_4 and t_7 are occurrence-matched which satisfies the *BEP* condition. Similarly, t_{10}, t_{11}, t_{15}, t_{16} and their descendants are pruned because of *BEP* criterion.

PROPOSITION 2. **(Forward Extension Pruning)** *For a k-ee subtree pattern t, if there exists a supertree $t' \in B_t^r$ such that t and t' are occurrence-matched and the parent of $t' \backslash t$ is v (where v is a vertex on the rightmost path of t), then neither t nor any supertrees of t as extensions of any proper ancestor node of v can be closed.*

For example, pattern t_6 and its descendants in Figure 5 are pruned since all conditions for FEP are satisfied as follows: (i) t_4 is in the right-blanket of t_3 (ii) D is $t_4 \backslash t_3$ (iii) A is the proper ancestor node of D's parent node in t_4 (iv) t_6 is an extension of t_3 by adding a node E at A. Similarly, pattern t_9 and their descendants are pruned because of the *FEP* criterion.

BEP described in Proposition 1 means that once we find a pattern t' is in the left-blanket of t that occurrence-matches with t, then we do not have to perform pattern-growth of t, because a pattern extension in the pattern-growth approach is performed in a depth-first traversal manner. *FEP* described in Proposition 2 is a simple corollary of the *BEP*.

Algorithm 1 and 2 describe how to incorporate the pruning methods *BEP* and *FEP* into pattern-growth approach to mine closed and frequent *k-ee* subtree patterns. In Algorithm 2, line 5 and 8 ensures the algorithm to work in a pattern-growth way. We check *BEP* condition at line 1, and *FEP* condition at line 6. In this way, we do not have to generate all frequent *k-ee* subtree patterns to mine closed patterns.

5. DISCRIMINATIVE K-EE SUBTREE PATTERN SELECTION

In Section 3, we developed an algorithm to mine closed and frequent *k-ee* subtree pattern, but there may still be too many resulting patterns. In this section, we present how to carefully select discriminative patterns from among the closed and frequent patterns in order to reduce the size of the feature set and to improve the performance of the classifier.

Based on the study that the patterns with high *Fisher* score can help improving the classification performance [7], we use it in our study to evaluate the discriminative power of a *k-ee* subtree pattern. The *Fisher* score is defined as

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

where n_i is the number of data samples in class i, μ_i is the average pattern frequency in class i, σ_i is the standard deviation of the pattern frequency in class i, and μ is the

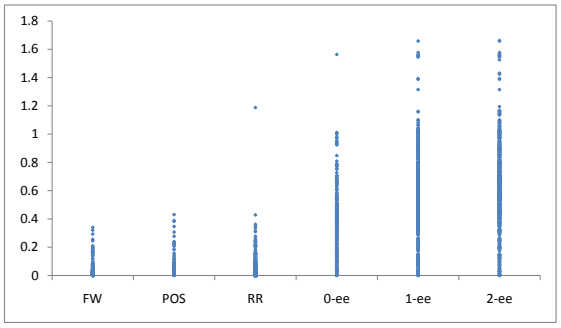

Figure 7: Fisher score distribution of various feature sets

Algorithm 3: Procedure *WSMine* to mine discriminative k-ee subtree patterns

input : Tree data set $\mathcal{D}$, min_sup θ
output: Discriminative k-ee features F of $\mathcal{D}$

$C \leftarrow$ `ClosedMine`($\mathcal{D}$,θ);
while $(C \neq \emptyset)$ *or* $(\mathcal{D} \neq \emptyset)$ **do**
  Select top-1 discriminative pattern t from C;
  $F \leftarrow F \cup \{t\}$;
  $\mathcal{D} \leftarrow \mathcal{D} - \{$trees that are covered by δ features in $F\}$;
  $C \leftarrow C - \{t\}$;
end

average pattern frequency in the whole dataset. A pattern will have a large *Fisher* score if it has similar values within the documents of the same class and very different values across the documents of different classes, at the same time.

Figure 7 presents *Fisher* score distributions of various feature sets such as function words (FW), POS tags (POS), rewrite rules (RR), and k-*ee* subtree patterns for k=0, 1, and 2 (0-ee, 1-ee, and 2-ee, respectively). We can easily see that the highest scores are mostly from k-*ee* subtree patterns, which implies that they can be more meaningful than other features. In fact, in the experiments, our k-*ee* subtree patterns achieved highest accuracy for all datasets.

Based on this *Fisher* score measure, we perform the feature selection procedure in a sequential coverage way as follows. We describe this procedure in Algorithm 3. We select the top scored pattern which covers at least one syntactic tree of the dataset and remove it from the list of the patterns. Moreover, any syntactic tree that is covered by at least δ features will be removed from the dataset. Here, *delta* is a feature coverage threshold introduced in [7]. It allows multiple patterns to represent a tree, which is known to improve the classification accuracy. Third, we go back to the second step until either the dataset becomes empty or no more patterns are left.

Once the feature selection procedure is complete, we get a small number of discriminative, closed, and frequent k-*ee* subtree patterns. Considering these patterns as a feature set, we express a document as a vector representation assigning a feature value by the frequency of the pattern described in Definition 4, and learn a classification model.

6. EXPERIMENTS

In this section, we present empirical evaluation results to validate the performance of our authorship classification framework. In particular, we conduct experiments on news articles and movie reviews. The experiments are designed to test whether our k-*ee* subtree patterns, as a new feature set, are useful for authorship classification.

Table 1: Statistics of News Articles

	# of documents	# of sentences	# of words
N1	100	3710	77501
N2	100	2666	59745
N3	100	5587	114337
N4	100	7198	129867
Total	400	19161	381450

Table 2: Statistics of Movie Reviews

	# of documents	# of sentences	# of words
M1	578	16508	423749
M2	567	15108	414295
M3	597	15320	357301
M4	415	4150	104337
Total	2177	51086	1299682

6.1 Datasets

We collected two different kinds of documents from a public data collection *The New York Times*: news articles and movie reviews. We got four authors with 400 documents for news articles, and four authors with around 2,000 documents for movie reviews.

For the news articles, we chose two journalists Eric Dash (N1) and Jack Healy (N2) from business department, and two other journalists Denise Grady (N3) and Gina Kolata (N4) from health department, who were one of the main contributors in their departments. The reason we collected documents in this way is because the journalists in the same department are likely to write articles in the same topic and genre using similar words. The statistics of each journalist are shown in Table 1.

For the movie reviews, we chose four main movie critics of *The Times*: A. O. Scott (M1), Manohla Dargis (M2), and Stephen Holden (M3), and Jeannette Catsoulis (M4). The reason we collected this data is because movie reviews of the same movie are likely to be in the same topic and genre using similar words. The statistics of each critics are shown in Table 2.

6.2 Evaluation Methodology

To evaluate the performance, we paired the authors of each domain and conducted binary classification on these 12 different datasets. For each dataset, we conducted 5-fold cross validation, and averaged the accuracy as a measure of the performance. For each fold, training data was used to mine the syntactic features and to get a classification model while test data was only used for prediction purpose. In this way, our evaluation ensured that there is no information leak from the test data for the classification task.

To show how effectively our new feature set works, we compared the authorship classification performance with other syntactic features such as function words, *POS* tags, and rewrite rules. As for function words, we took the list of 308 function words from [21]. We used 70 *POS* tags generated

Table 3: Number of Features

Domain	RR	0-ee	1-ee	2-ee
News Articles	3929	280.83	560.23	789.93
Movie Reviews	9029.2	557.87	1348.9	2074.5

from the stanford parser [18]. The number of features of the other feature sets are presented in Table 3. For each feature set and for each dataset, we computed the average value of the number of distinct features of 5-fold training data. In the table, we showed the average of the number of distinct features for each domain. The difference of the number of features between different domains implies that movie reviews are written in more sophisticated way than news articles. That also implies indirectly that it would be harder to classify movie reviews than news articles. We see that rewrite rules are using the biggest number of features.

We used the occurrences of each feature as a feature value for the syntactic features except k-ee subtree patterns which used a new frequency measure defined in Definition 4. For the fair comparison, we used the same classifier, linear-kernel *SVM* (with the parameter tuned for the best performance of each feature set), which was previously shown to work reliably with high accuracy on authorship classification [11].

6.3 Performance Evaluation

Authorship classification accuracies for various feature sets are presented in Table 4 and 5. All experimental results of k-ee subtree pattern-based classification used (relative) minimum support threshold 0.1 for frequent pattern mining and sequential coverage threshold 10 for discriminative pattern mining by default. We can easily find that our proposed feature set of k-ee subtree patterns, especially for $k = 1, 2$, achieved the highest accuracies for most of the datasets. We see that using embedded edges can help to enhance the authorship classification performance, but we cannot say more embedded edges would get better performance. For a higher number of embedded edges (k), even we utilize several pruning techniques, it is intractable to mine them all. Moreover, higher k sometimes tends to overfit to training data that might degrade the accuracy performance. We can conclude that a small number of embedded edges is enough to achieve high performance of classification task for both in accuracy and efficiency aspects.

For both data collections of news articles and movie reviews, all feature sets showed similar tendencies. 1-ee and 2-ee showed the highest accuracies of news article datasets and movie review data collections repectively, while POS got the worst accuracies for both data collections. Among 12 datasets of experiments, N12, N34 and M12 showed bad performances for all feature sets. Analyzing statistics of data collections in Table 1 and 2, we see that the classes in them has similar number of words which indirectly shows those classes are hard to classify. Especially for dataset N34, both classes of N3 and N4 are from health department of news domain and they have quite a few quotations with informal style of writings which made it the hardest dataset to be classified. It is noticeable that even for this hard dataset, our feature set got the highest accuracy with a big gap of performance to the other feature sets.

We calculated the standard deviation of the accuracies to show how reliable the feature sets are, and found that k-ee subtree patterns achieved consistent results for both data collections.

Overall, we conclude that k-ee subtree patterns are meaningful features for authorship classification which works reliably for real life data collections and achieves high accuracy.

Table 4: Accuracy Comparisons (News Articles)

	FW	POS	RR	0-ee	1-ee	2-ee
N12	91.5	87	94	**96**	95	95.5
N13	94	85	91	97.5	**98**	97.5
N14	95.5	92.5	96	94.5	**96.5**	95
N23	95	92.5	92.5	96.5	98.5	**99**
N24	97	95.5	97.5	**98.5**	**98.5**	**98.5**
N34	80.5	67.5	67.5	88.5	**90**	**90**
AVG	92.3	86.7	89.8	95.3	**96.1**	96.0
STD	6.04	10.16	11.15	3.57	**3.28**	3.31

Table 5: Accuracy Comparisons (Movie Reviews)

	FW	POS	RR	0-ee	1-ee	2-ee
M12	92.8	81.0	88.0	92.48	**94.26**	94.22
M13	93.6	92.5	92.7	95.22	95.06	**95.8**
M14	92.1	88.0	94.2	97	97.4	**97.7**
M23	94.4	92.8	94.8	97.58	**97.92**	97.58
M24	93.1	91.0	92.9	95.22	96.04	**96.32**
M34	93.1	88.6	94.9	97.12	**97.22**	97.12
AVG	93.2	89.0	92.9	95.8	96.3	**96.5**
STD	*0.77*	4.40	2.59	1.90	1.45	1.32

7. CONCLUSION

In this paper, we proposed a novel solution for an authorship classification problem by mining discriminative closed *k-ee* subtree patterns. First, we designed a new feature set of *k-ee* subtree patterns which contains more meaningful syntactic structures of a sentence than previous feature sets which are based on simple forms of syntactic features including function words, *POS* tags, and rewrite rules. To mine *k-ee* subtree patterns, we developed a closed frequent *k-ee* tree mining algorithm by use of several pruning techniques. We performed a *Fisher* score based feature selection procedure on top of those mined patterns. This small set of discriminative patterns could effectively classify the documents based on their authorship.

Experimental study has been performed on two real datasets, news articles and movie reviews, from *The New York Times* public data corpus. These data collections were carefully chosen to ensure to be in the same genres using similar terms. Our *k-ee* subtree pattern based classification achieved the best results compared to other feature sets such as function words, *POS* tags, and rewrite rules.

In future research, we want to develop a way to directly mine discriminative *k-ee* subtree patterns, not generating all closed patterns. Usually, discrimiative patterns selected from closed patterns with low minimum support threshold θ show better accuracy, but it is hard to find an optimized θ since the mining cost increases exponentially when θ becomes lower. A directive way of mining discriminative pattern might work without specifying θ which would guarantee high quality of discriminative patterns.

8. REFERENCES

[1] S. Argamon and S. Levitan. Measuring the usefulness of function words for authorship attribution. In *ACH/ALLC*, 2005.

[2] S. Argamon, M. Šarić, and S. S. Stein. Style mining of electronic messages for multiple authorship discrimination: first results. In *KDD*, 2003.

[3] H. Baayen, H. van Halteren, and F. Tweedie. Outside the cave of shadows: using syntactic annotation to enhance authorship attribution. *Literary and Linguist Computing*, 11(3):121–132, 1996.

[4] S. Bloehdorn and A. Moschitti. Structure and semantics for expressive text kernels. In *CIKM*, 2007.

[5] B. Bringmann and A. Zimmermann. Tree^2 - decision trees for tree structured data. In *PKDD*, 2005.

[6] S. Burrows, A. L. Uitdenbogerd, and A. Turpin. Application of information retrieval techniques for source code authorship attribution. In *DASFAA*, 2009.

[7] H. Cheng, X. Yan, J. Han, and C.-W. Hsu. Discriminative frequent pattern analysis for effective classification. In *ICDE*, 2007.

[8] H. Cheng, X. Yan, J. Han, and P. S. Yu. Direct discriminative pattern mining for effective classification. In *ICDE*, 2008.

[9] Y. Chi, Y. Xia, Y. Yang, and R. R. Muntz. Mining closed and maximal frequent subtrees from databases of labeled rooted trees. *IEEE Transactions on Knowledge and Data Engineering*, 17(2):190–202, 2005.

[10] O. de Vel, A. Anderson, M. Corney, and G. Mohay. Mining e-mail content for author identification forensics. *SIGMOD Record*, 30(4):55–64, 2001.

[11] J. Diederich, J. Kindermann, E. Leopold, and G. Paass. Authorship attribution with support vector machines. *Applied Intelligence*, 19(1-2):109–123, 2003.

[12] M. Gamon. Linguistic correlates of style: authorship classification with deep linguistic analysis features. In *COLING*, 2004.

[13] A. M. García and J. C. Martín. Function words in authorship attribution studies. *Literary and Linguistic Computing*, 22(1):49–66, 2007.

[14] J. Grieve. Quantitative authorship attribution: An evaluation of techniques. *Literary and Linguistic Computing*, 22(3):251–270, 2007.

[15] G. Hirst and O. Feiguina. Bigrams of syntactic labels for authorship discrimination of short texts. *Literary and Linguistic Computing*, 22(4):405–417, 2007.

[16] D. L. Hoover. Statistical stylistics and authorship attribution: an empirical investigation. *Literary and Linguistic Computing*, 16(4):421–444, 2001.

[17] D. L. Hoover. Another perpective on vocabulary richness. *Computers and the Humanities*, 37(2):151–178, 2003.

[18] D. Klein and C. D. Manning. *The Standford parser: A Statistical Parser*, 2002. `http://nlp.stanford.edu/software/lex-parser.shtml`.

[19] D. Lo, H. Cheng, J. Han, S.-C. Khoo, and C. Sun. Classification of software behaviors for failure detection: a discriminative pattern mining approach. In *KDD*, 2009.

[20] T. C. Mendenhall. Spelling checkers,spelling correctors and the misspellings of poor spellers. *Science*, 11(214):237–246, 1887.

[21] R. Mitton. Spelling checkers,spelling correctors and the misspellings of poor spellers. *Information Processing and Management*, 23(5):495–505, 1987.

[22] A. Moschitti. Efficient convolution kernels for dependency and constituent syntactic trees. In *ECML*, 2006.

[23] F. Mosteller and D. L. Wallace. *Inference & Disputed Authorship: The Federalist*. Addison Wesley, 1964.

[24] J. Pei, J. Han, B. Mortazavi-asl, H. Pinto, Q. Chen, U. Dayal, and M. chun Hsu. Prefixspan: Mining sequential patterns efficiently by prefix-projected pattern growth. In *ICDE*, 2001.

[25] J. Rudman. The state of authorship attribution studies: Some problems and solutions. *Computers and the Humanities*, 31(4):351–365, 1998.

[26] C. Sanderson and S. Guenter. Short text authorship attribution via sequence kernels, markov chains and author unmasking: an investigation. In *EMNLP*, 2006.

[27] E. Stamatatos. A survey of modern authorship attribution methods. *Journal of the American Society for Information Science and Technology*, 60(3):538–556, 2009.

[28] A. Termier, M.-C. Rousset, M. Sebag, K. Ohara, T. Washio, and H. Motoda. Dryadeparent, an efficient and robust closed attribute tree mining algorithm. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 20(3):300–320, 2008.

[29] J. Wang and J. Han. Bide: Efficient mining of frequent closed sequences. In *ICDE*, 2004.

[30] K. Wang, Z. Ming, and T.-S. Chua. A syntactic tree matching approach to finding similar questions in community-based qa services. In *SIGIR*, 2009.

[31] X. Yan, H. Cheng, J. Han, and P. S. Yu. Mining significant graph patterns by leap search. In *SIGMOD*, 2008.

[32] X. Yan and J. Han. Closegraph: mining closed frequent graph patterns. In *KDD*, 2003.

[33] M. J. Zaki. Efficiently mining frequent trees in a forest. In *KDD*, 2002.

[34] M. J. Zaki and C. C. Aggarwal. Xrules: an effective structural classifier for xml data. In *KDD*, 2003.

[35] Y. Zhao and J. Zobel. Effective and scalable authorship attribution using function words. In *AIRS*, 2005.

[36] Y. Zhao, J. Zobel, and P. Vines. Using relative entropy for authorship attribution. In *AIRS*, pages 92–105, 2006.

[37] R. Zheng, J. Li, H. Chen, and Z. Huang. A framework for authorship identification of online messages: Writing-style features and classification techniques. *Journal of the American Society for Information Science and Technology*, 57(3):378–393, 2006.

[38] A. Zimmermann and B. Bringmann. Ctc — correlating tree patterns for classification. In *ICDM*, 2005.

[39] L. Zou, Y. Lu, H. Zhang, R. Hu, and C. Zhou. Mining frequent induced subtrees by prefix-tree-projected pattern growth. In *WAIMW*, 2006.