

Recursive Data Mining for Author and Role Identification

Vineet Chaoji, Apirak Hoonlor and Boleslaw K. Szymanski
 {chaojv, hoonla, szymanski}@cs.rpi.edu

Department of Computer Science and Center for Pervasive Computing and Networking
 Rensselaer Polytechnic Institute, Troy, NY, 12180

Abstract — Like paintings and verbal dialogues, written documents exhibit the author's distinctive style and identification of the author of an anonymous document is an important and challenging task in computer security. Even more challenging is identification of a style of a group of diverse individuals acting in similar circumstances, like authors writing in certain literary period or people writing in a certain social role. The last application is important for analyzing hidden group communicating over the internet in which neither identities nor roles of the members are known. Other applications of the identification of such styles include fraud detection, author attribution and user profiling. The task of finding distinctive features of an artifact has much broader scientific implications that range from art and scriptures to network security.

In this paper, we focus on capturing patterns in electronic documents. The approach involves discovering patterns at varying degrees of abstraction, in a hierarchical fashion. The discovered patterns capture the stylistic characteristics of either the author, or a group of authors, or even of the specific role that the author plays in relation to others. These patterns are used as features to build efficient classifiers. Due to the nature of the pattern discovery process, we call our approach *Recursive Data Mining*. The patterns discovered allow for certain degree of approximation, which is necessary for capturing non-trivial patterns on realistic datasets. Experiments on the Enron and SEA datasets, the former categorizes members into organizational roles and the latter categorizes a set of computer sessions, are conducted to substantiate our methodology. The results show that a classifier that uses the dominant patterns discovered by Recursive Data Mining performs better than the same classifier without features based on RDM patterns, in role detection and author identification.

Index Terms— Data mining, feature extraction, text mining, pattern discovery.

I. INTRODUCTION

IN recent years the internet has dramatically increased its presence in our day to day activities. From online gaming and shopping to meeting prospective life partners, the online world has captured every aspect of our life. The emergence of weblogs and social networking (online community) sites (e.g., MySpace and Facebook) has enabled people to connect across

countries and continents. This has not only reduced the degree of separation between people but also allowed them to collaborate with a larger audience. On the other hand, these online communities provide a safe haven for malicious activities by rogue users. Weblogs too induce online communities by allowing users to share their views and opinions to a large audience base. Consequently, they too have been seen as a propaganda media as well as a breeding place for malicious and covert activities [20]. This massive reach and impact of online media has provided a thriving ground for internet-based terrorism and espionage, where sheer size of the community and its interactions is used to hide the group existence and on-line activities [20]. Identifying such malicious individuals and hidden groups of users is an important task for curtailing cyber-crime and preserving online privacy. Nevertheless, the task is difficult given the size of such social networks and the amount of data involved. Previous efforts [3] have used the structure of the connection network of online communities to identify such groups of users. Few efforts have been directed towards characterizing groups of users based on the content generated within the online communities and weblogs. The availability of large volumes of data - messages between users, blog posts and email communication - provides the necessary impetus for this direction. This text data contains ample clues (referred to as *patterns* or *features* in the rest of the paper) for us to infer the source of the text.

The task of identifying the author of a document is commonly termed as the *author identification problem*. Even more difficult, yet important for hidden group operating over the internet or inside a legal organization is identifying the role or the relation of the sender to the receiver based on their direct emails or messages. This task is called *role identification*. Role identification can be seen as a generalization of the author identification task wherein common characteristics of a group of authors need to be identified. Role identification can help in identifying the relationship between agents within a group, which in turn can help decipher the structure of a group. This can potentially help uncover the roles of different agents in an organized terrorist activity.

In this paper, we present a general pattern extraction

method, termed *Recursive Data Mining* (RDM). The basic premise behind our approach is that within any form of communication, traces of personal style can be identified. Moreover, every person employs multiple styles based on the stature or relationship to the recipient of the communication. For instance, the manner in which a person communicates with his/her superior is quite different from the manner in which one would communicate to his/her friend. Or the manner in which a person writes to his/her parents is different from the way he/she writes to his children. The interesting question is as follows. Are individual style differences between senders in the same role so large that the individual style variations would mask their roles? For example, is a thank you note from a rude person distinguishable from a complain letter of a polite person? Hence, in this paper we want to demonstrate (through our results) that the stylistic characteristics of a certain role overpower the individualistic characteristics of a person.

To address the above mentioned challenges RDM uses an approach that extracts syntactic patterns. These patterns capture the stylistic characteristics, which are in turn used to attribute an authorship or a role to an individual. Within the machine learning community, the term feature extraction is commonly used for techniques that identify relevant features (patterns in our case) for a given application. The term feature can represent keywords for text documents or principle eigenvectors for high dimensional genetic data. Feature extraction is broadly considered to be composed of two sub-tasks – *feature construction* and *feature selection* [6]. Presence of noise in the data hinders the task of identifying meaningful signals, which in turn causes the feature construction stage to return ineffective features. Methods for selecting relevant features fall under the feature selection task. Many key applications in computer and network security have considerably utilized and benefited from feature extraction. Smart feature selection is used for both intrusion detection [2, 10] and compression of data to minimize network traffic [18].

The input data is treated as a sequence of tokens. In any form of sequence-based information, traces of personal styles can be identified by authorship analysis (see [12], [10], and [16]). The RDM framework discovers statistically significant sequence patterns from a stream of data. The approach is independent of any semantic information making it amenable for text documents in different languages, and can be applied to sequence-based data of any nature (time series data, genome data, etc.). The method also provides a certain degree of flexibility by allowing gaps in the patterns. Gaps remove the restriction of exact matches between patterns by acting as a wildcard character (see Section 4), thus enabling approximate matches. RDM for role identification has been initially described in [4] as part of a set of tools for social network analysis. In this paper, we argue that applications in the area of security and forensics have data that can be represented as a sequence of tokens. Furthermore, these applications can be formulated as either the author identification task or the role detection problem in our framework.

Some of the key technical contributions of this work are:

- The patterns formed do not have any length restriction. This allows arbitrary size patterns to be discovered. Most of the other published techniques work on a fixed size window.
- Statistical techniques, such as log likelihood ratio tests, permutation tests and score-based sequence analysis methods are used for identifying significant patterns.
- The method is hierarchical in nature. This enables us to capture patterns at various levels of abstractions. Moreover, the hierarchical nature allows us to remove noisy symbols from the stream as we move from a lower level to a higher level in the hierarchy. This ultimately leads to discovery of *long range patterns* that are separated by long noisy intermediate segments.
- The method is also able to discover approximate (similar) patterns.

The rest of the paper is organized as follows. Section 2 discusses related work. Section 3 introduces the basic terminology for this work. Section 4 provides a detailed description of our methodology. The experimental results are presented in Section 5.

II. RELATED WORK

Even though the proposed technique is useful for identifying informal groups (internet chat groups, blogger groups, etc.) in web communication, the approach is general enough to be applied to other areas such as masquerade detection [13] and intrusion detection [11]. In the area of computer intrusion, many previous efforts have relied heavily on feature extraction methods from sequential input. Schonlau et al. [15] experimented on a sequence match method on a data set based on UNIX user truncated commands for masquerade detection task. The results from [15] illustrate that a simple sequence match does not perform well. Seo and Cha [16] introduced a combination of SVM and sequence-based user commands profile on Schonlau’s dataset. The results show significant improvement over previous work. Hierarchical approach has been applied on the same dataset by Szymanski et al. [19] using RDM algorithm. Instead of using sequential-base features, in [19], RDM was applied to extract statistical information regarding the sequence, such as number of distinct tokens, number of “dominant” patterns, etc. We would like to point out that the work presented in this paper is much enhanced as compared to the initial RDM framework presented in [19]. In [19], statistics over the entire data are used to build a single classifier; in contrast, this work uses patterns as features and it trains an ensemble of classifiers as compared to a single classifier in [19]. Also, in [19], no gap is allowed in a pattern.

In other areas, previous efforts have explored the benefits of hierarchical approach for analyzing the underlying structure of text documents [14], [17]. In [14] the authors extract a hierarchical nested structure by substituting grammar for repeated occurrences of segments of tokens. Similarly, in [17]

the authors present a data independent hierarchical method for inferring significant rules present in natural languages and in gene products. Our efforts differ in that we provide certain flexibility in the patterns found by allowing gaps. This enables us to work with much smaller datasets as compared to [17]. From the experiments, we also show that the hierarchical sequence model can catch certain mannerism in human language and that it is an extension of the bag-of-words approach. The dominant patterns generated might strike a resemblance with frequent closed sequence patterns, but the significance test in RDM is “smarter” than a simple frequency check. Moreover, patterns with gaps are different from both embedded and induced sequence patterns. Identifying patterns using significance tests was used extensively in biological sequence analysis [8, 9]. In recent work [1], the authors modify the frequency-based mining task, to obtain useful patterns which are in turn used for classification.

III. PRELIMINARIES

Consider a set of sequences, denoted as SEQ . Each sequence consists of a series of **tokens** from a set \mathcal{T} . A sequence $S \in SEQ$ can be represented as t_1, t_2, \dots, t_n , where $t_i \in \mathcal{T}$ and n is the length of the sequence. Based on the application, a token can represent a different entity. For instance, within text documents, a token can either represent a character or a word and a sequence S would then correspond to the whole document. For stock market data, each token could represent a pair of numeric values (price and volume) and the sequence represents the entire time series of purchases (or sales) of a certain stock. A special token, called the **gap token**, corresponds to a blank entry and is represented by the symbol \perp . The gap token mimics the ‘.’ character in regular expressions - it can be matched with any other token. A **sequence pattern** \mathcal{P} is an ordered sequence of tokens from $\mathcal{T} \cup \{\perp\}$. Formally, \mathcal{P} can be denoted as $\{s_i: s_1 \in \mathcal{T} \text{ AND } s_i \in \mathcal{T} \cup \{\perp\}, i = 2 \dots p_l\}$, where i is the index of a token in the sequence and p_l is the length of the pattern. Note that the first and last tokens are never the gap token. This condition is useful for combining contiguous patterns. In the rest of the paper, the term pattern would always imply a sequence pattern and terms pattern and feature would be used interchangeably, unless stated otherwise.

Two patterns are said to have an **exact match** if they consist of the same sequence of tokens. Given a **similarity function**, $sim(\mathcal{P}_1, \mathcal{P}_2)$ a similarity score is assigned to each pair of patterns. Exact matching restricts the similarity score to binary values - $sim(\mathcal{P}_1, \mathcal{P}_2) = 1$ if $\mathcal{P}_1 = \mathcal{P}_2$, 0 otherwise. The presence of a gap token in a sequence pattern relaxes the exact match constraint, allowing it to match a wider set of patterns with $sim(\mathcal{P}_1, \mathcal{P}_2) \in [0, 1]$. A match with similarity score greater than $\alpha \in (0, 1)$ is called a **valid match**. The set $\mathcal{M}_\mathcal{P}$ is the **set of valid matches** for a pattern \mathcal{P} . A pattern \mathcal{P} of length l and g gaps is termed as a **(l, g)-pattern**. If \mathcal{P} has a match at index i in sequence S , then it belongs to the set of patterns $S_i(l, g)$ -

patterns. The set of patterns $S_i(l)$, given by the expression $\bigcup_{g=0}^{\max_gap} S_i(l, g)$ represents all patterns of length l starting at index i in S . \max_gap , as the name indicates, is the maximum number of gaps allowed in a pattern.

IV. RECURSIVE DATA MINING

Recursive Data Mining (RDM) is an approach for discovering features from sequences of tokens. Given a set of sequences as input, the algorithm captures statistically significant patterns from the initial sequences. The patterns obtained are assigned new tokens. The initial sequences are re-written by collapsing each sequence pattern to its newly assigned token, while retaining the rest of the tokens. The algorithm now operates on the re-written sequences and continues to iterate through the pattern generation and sequence re-writing steps until either the sequences cannot be re-written further or a predefined number of iterations is reached. Each generation of sequences in the above process is termed a level, with the initial set of sequences called *level-0 sequences*. The patterns obtained at each level form a set of features. The term “recursive” in the title refers to this iterative step that obtains the next level by operating on the current level. In the RDM process, we claim that the recursive (hierarchical) processing of the data captures distinctive features at varying levels of abstraction. Intuitively, at lower levels the patterns obtained are more specific; resulting in a smaller set of valid matches (\mathcal{M}). At higher levels, the patterns are more general, resulting in a larger \mathcal{M} set. On the other hand, with increasing levels, the number of patterns found decrease monotonically.

In this section we present the details of an RDM based classifier. Like most supervised learning tools, RDM has two stages of processing – training and testing. The *training phase* starts with *pattern generation*, followed by pattern selection through the *pattern significance* step. Out of the significant patterns, the *dominant patterns* form the feature set for a level.

The overall RDM process is outlined in Algorithm 1. The input is a set of sequences SEQ_{INIT} , which also forms the initial

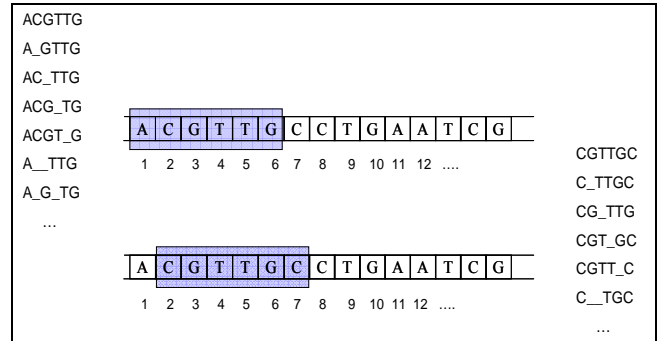


Fig 1. Pattern Generation Step. ($l = 6, \max \text{ gap} = 2$)

set of sequences SEQ_0 in the iterative procedure. The set of sequences for level $(i+1)$ are generated from the sequences in level i and the set of dominant patterns \mathcal{D} . \mathcal{P}_{ALL} and \mathcal{P}_{SIG} represent the sets of all and significant patterns respectively.

Dominant patterns (denoted by \mathcal{D}) for a level are obtained from the *get_domi_patterns* method. The union of dominant patterns at each level is collected in \mathcal{L} .

Algorithm 1 Recursive Data Mining

Input: Set of sequences SEQ_{NIT}
Output: Sets of patterns (features) \mathcal{L} , one for each level

```

1:  $\mathcal{L} = \{\}$ 
2:  $i = 0$ 
3: repeat
4:   if  $i == 0$  then
5:      $SEQ_i = SEQ_{NIT}$  // Level 0
6:   else
7:      $SEQ_i = make\_next\_level(SEQ_{i-1}, \mathcal{D})$  // Level  $i$ 
8:   end
9:    $\mathcal{P}_{ALL} = pattern\_generation(SEQ_i)$ 
10:   $\mathcal{P}_{SIG} = sig\_patterns(SEQ_i, \mathcal{P}_{ALL})$ 
11:   $\mathcal{D} = get\_domi\_patterns(SEQ_i, \mathcal{P}_{SIG})$ 
12:   $\mathcal{L} = \mathcal{L} \cup \mathcal{D}$ 
13:   $i++$ 
14: until  $\mathcal{D} == \emptyset$  or  $i == max\_level$ 
15: return  $\mathcal{L}$ 

```

A. Pattern Generation

Each input instance is converted into a sequence of tokens. The initial sequence of tokens, *level-0 sequence*, will be referred to as S_0 . A sliding window of length l_w moves over S_v ($v = 0$ initially). At each position p of the window, all possible (l_w, max_gap) -sequence patterns are generated. The number of patterns generated equals the number of combinations of tokens covered by the window along with the gap token. Since we do not allow a gap for token s_1 and s_{max_gap} of a pattern, the number of patterns generated at each window position is given by $\sum_{g=0}^{max_gap} \binom{l_w - 2}{g}$. So, for a $l_w = 5$, with $max_gap = 2$, seven patterns are generated. A bounded hash keeps count of the number of occurrences of each pattern at level v , as the sliding window moves over S_v . This forms the first pass over sequence S_v . Figure 1, shows the patterns generated at position 1 and 2 of the sequence.

B. Pattern Significance

The number of (l_w, max_gap) -patterns uncovered in the sequences is generally large. Many of those patterns are either very specific to a certain sequence or insignificant because they contain commonly occurring tokens. In either case, they are ineffective in capturing any stylistic attributes while adding to the computation cost of the algorithm. The “usefulness” of a pattern is computed with a statistical significance test. Patterns

that are deemed insignificant are eliminated from further consideration. For a set of sequences SEQ , let the unique tokens in the entire set be denoted by \mathcal{T} . The frequency of a token t_i in SEQ is denoted by f_{t_i} . So the probability of token t_i

over SEQ is $\frac{f_{t_i}}{|\mathcal{T}|}$. For a pattern \mathcal{P} of length l_w , the

probability of tokens in the pattern can be represented as $(p_{t_1}, p_{t_2}, \dots, p_{t_{l_w}})$. Note that gaps are considered as special tokens. The probability of pattern \mathcal{P} is thus given by the expression

$$\mathbf{P}(\mathcal{P}) = \mathbf{P}(RV_1=t_1, RV_2=t_2, \dots, RV_{l_w}=t_{l_w}) \quad (1)$$

$$= p(t_1)p(t_2|t_1) \cdots p(t_{l_w}|t_1, \dots, t_{l_w-1})$$

where RV_i is a random variable for the token t_i . Assuming that the words are independent of each other (this assumption is just for the purpose of measuring pattern significance, because if they are not, we will eventually catch their relationship at the higher level of RDM abstraction), just the marginal probabilities for the words need to be computed resulting in

$$\mathbf{P}(\mathcal{P}) = \prod_{i=1}^{l_w} p_{t_i} \quad (2)$$

The probability of a gap token is ϵ , which is a user defined parameter (see Section *Dominant Patters* for details). The probability of occurrence of \mathcal{P} under the random and independent assumption is given by

$$\mathbf{P}_R(\mathcal{P}) = \mathbf{P}(RV_1=t_1, RV_2=t_2, \dots, RV_{l_w}=t_{l_w}) \quad (3)$$

Since each token is assumed to be equally likely under the random assumption, the above expression simplifies to $\mathbf{P}_R(\mathcal{P}) =$

$$\left(\frac{1}{|\mathcal{T}|} \right)^{l_w}$$

The ratio $\mathbf{P}_R(\mathcal{P})/\mathbf{P}(\mathcal{P})$ is used to determine significance of the pattern. If the above ratio is smaller than 1, then the pattern is considered to be significant, otherwise it is considered insignificant. The ratio indicates the likelihood of the pattern to occur under the random model as compared to its occurrence under the unknown observed distribution. This is similar in essence to the log-likelihood ratio test, with null hypothesis (H_0), that the observed distribution is similar to the random distribution. The alternate hypothesis H_1 states otherwise. The log-likelihood ratio is given by the expression

$$\mathbf{LRT} = -2\log_e(\mathcal{L}_R(\theta)/\mathcal{L}_O(\theta)) \quad (4)$$

where $\mathcal{L}_R(\theta)$ is the likelihood function under the random model and $\mathcal{L}_O(\theta)$ is the likelihood for the observed distribution. H_0 is a special case of H_1 , since it has fewer parameters (captured by θ) as compared to the more general alternate hypothesis. Applying the significance test to the set of patterns \mathcal{P}_{ALL} gives us a smaller set of significant patterns, \mathcal{P}_{SIG} . In practice, computational cost of the pattern generation step can be reduced by checking whether a sequence of tokens in the current window have the ratio of $\mathbf{P}_R(\mathcal{P})/\mathbf{P}(\mathcal{P})$ greater than 1 or not. If not, then we can conclude that no pattern generated from this current window can be significant. Hence, we can

reduce the computation cost for generating all possible patterns from this sliding window.

Other significance tests for sequence patterns have been proposed. Permutation tests [5] provide a simple approach for comparing the observed occurrences of a pattern with the number of likely occurrences over a random sequence. The practical application of this method requires generating a large number of random permutations of the input sequence and computing the statistics on the random permutations. If the input sequence is long, this operation can be computationally very expensive. Karlin et al. [8], [9] have proposed many significance tests for identifying relevant regions in protein sequences. Their approach relies on assigning scores to the tokens such that the sum of the expected scores for all the tokens is negative. Such conditions are easier to find for biological sequences as compared to text documents.

C. Dominant Patterns

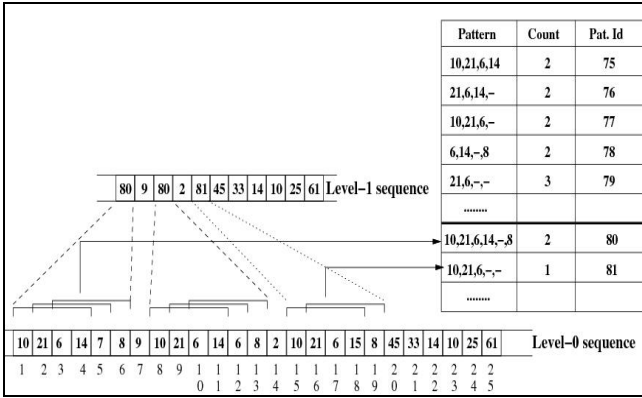


Fig. 2. Sequence Re-writing Step

After the significant patterns at level v are determined, a second pass is made over the sequence of tokens S_v . At each position in the sequence, the tokens in the significant patterns are matched against the tokens in the sequence. The matching score is defined as the conditional probability of a match given two symbols, i.e., if $\mathcal{P}[i]$ and $S_v[j]$ are the same then the conditional probability of a match is 1. On the other hand, if $\mathcal{P}[i] = \perp$ then the conditional probability is ε . The matching score is based on the following rules

$$score(\mathcal{P}[i], S_v[j]) = \begin{cases} 1 & \text{if } \mathcal{P}[i] = S_v[j] \\ \varepsilon & \text{if } \mathcal{P}[i] = \perp, \varepsilon < 1, \\ 0 & \text{if } \mathcal{P}[i] \neq S_v[j] \end{cases}$$

where $\mathcal{P}[i]$ is the i^{th} token of the pattern and j is an index over sequence S . ε is intended to capture the notion that a \perp symbol is not as good as an exact match but much better than a mismatch. The value of ε is a user defined parameter and it is set to be greater than 0.5 in our experiments to favor a match with the gap token. The total score for a pattern, starting at index j in S , is given

$$\text{by } score(\mathcal{P}, S_v[j]) = \sum_{i=1}^{P_l} score(\mathcal{P}[i], S_v[j+i]). \text{ The pattern}$$

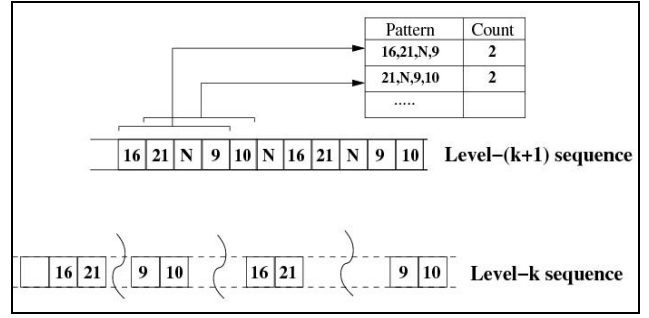


Fig. 3. Removing Noisy Tokens for Long Range Patterns

that has the highest score starting at location j in the input sequence is termed as the **dominant pattern** starting at position j . In other words, this is a pattern x defined by the expression $\text{argmax}_{x \in S_v} score(x, S_v[j])$. The term dominant pattern is coined from the fact that this pattern dominates over all other significant patterns for this position in the sequence. Two dominant patterns that are placed in tandem can be merged to form longer dominant patterns. The merging process is continued till no further dominant patterns can be merged. An example of the merging process is shown in Figure 2. A new token is assigned to each dominant pattern. During this second pass over the sequence at level v , the sequence for level $v+1$ is formed. The sequence corresponding to a dominant pattern is replaced by the new token for this dominant pattern. When a dominant pattern is not found at position j , the original token is copied from sequence S_v to the new sequence S_{v+1} . Figure 2 illustrates this step.

As the RDM algorithm generates subsequent levels, certain tokens get carried over from lower levels without participating in any dominant patterns at higher levels. Such tokens are termed “noisy” for the following reasons. First, they do not contribute to any patterns at these levels. Second, they obstruct the discovery of patterns that are separated by a long sequence of noisy tokens. Patterns separated by noisy tokens are called long range patterns. An example of a long range pattern is shown in Figure 3. These long range patterns can be captured only if the noisy tokens lying in between them can be collapsed. As a result, at each level, we collapse contiguous sequence of tokens that have not resulted in new dominant patterns for the last k levels, into a special noise token. Figure 3 illustrates the process of collapsing noisy tokens into a single special noise token N . Once the noisy tokens are collapsed, distant tokens may now fall within the same window, leading to more patterns being discovered at higher levels.

The set of dominant patterns \mathcal{D}_v for level v form the features for this level. This iterative process of deriving level $v+1$ sequence from level v sequence is carried on till no further dominant patterns are found or $v+1$ has reached a user predefined maximum value. The sets of features extracted are utilized by an ensemble of classifiers.

D. Training Phase

The training phase involves using dominant patterns generated at each level to construct an ensemble of classifiers $(C_1, C_2, \dots, C_{\text{max_level}})$, one for each level. The dominant patterns

used as features reflect the most relevant patterns, ignoring the highly frequent and infrequent patterns (upper and lower cut-offs). The upper and lower cut-offs are intended to prevent the use of insignificant patterns as features. The classifiers can be created from any machine learning method, such as Naïve Bayes or Support Vector Machine. Given a set of training sequences SEQ_{tr} , along with the labels r_1, r_2, \dots, r_m of all possible classes, dominant patterns are generated for each sequence starting at level 0 up to level max_level . The union of all tokens in \mathcal{T} and key dominant patterns at level v across all sequences in SEQ_{tr} forms the set of feature for classifier C_v . For the ensemble of classifiers, the final posterior class probability is the weighted sum of the class probabilities of individual classifiers. Each classifier is assigned a weight that reflects the confidence of the classifier. In order to determine this confidence value, the set SEQ_{tr} is further split into a training set SEQ_{new} and a tuning set. Each classifier in the ensemble trains its model based on SEQ_{new} . The accuracy of the classifier on the tuning set determines the confidence of classifier C_i ,

$$conf(C_i) = \frac{accuracy(C_i)}{\sum_{j=1}^{max_levels} accuracy(C_j)}.$$

E. Testing Phase

After the training phase discovers features from the training data, the testing phase finds occurrences of those features in the test data. The testing phase as such follows the training phase in terms of the level by level operating strategy. If a dominant pattern X was discovered at level-Y during the training phase, then it can be only applied to level-Y in the testing phase. Initially, the frequencies of tokens and level-0 dominant patterns are counted over the level-0 test sequence. This vector of frequencies forms the feature vector at level-0. Once the feature vector for level-0 is obtained, the next level sequence is generated. This is achieved by substituting the token of the best matching pattern at every position in the level-0 test sequence. Note that if the best match is less than a user specified threshold then the token at level-0 is carried over to level-1. Now the occurrences of the dominant patterns at level-1 are counted over level-1 test sequence. This process continues till all levels of dominant patterns are exhausted. Each classifier in the ensemble classifies the test data and the final probability is assigned based on the following weighing scheme

$$P(C | x) = \sum_{i=1}^{max_levels} conf(C_i) \times P_{C_i}(C | x) \quad (5)$$

where x is a test sequence and $P_{C_i}(C | x)$ is the posterior probability assigned by classifier C_i .

V. EXPERIMENT AND RESULTS

We applied RDM to two tasks: (1) author identification, and (2) role identification. For the author identification task, RDM was applied on the SEA dataset [15]. For role identification

task, RDM is applied to the Enron dataset. The detailed description of experimental setup and results for each task are presented in Section V-A and V-B respectively.

A. Author Identification Task

We illustrate that RDM using tokens and patterns as features performs as well as Naïve Bayes, Support Vector Machines, and RDM using statistical information as features [19]. We use SEA dataset [15] for this task.

1) Data Preparation and Experimental Setup

The SEA dataset consists of 50 files, where each file corresponds to one user. Each file contains 15000 commands. The first 5000 commands of each file do not contain any masqueraders, and is considered as training data. The other 10000 commands are seeded with masquerading users. For this experiment, we consider only the training data portion of the file. We split training data of each user into two sets: 2500 commands for training set and other 2500 commands for validation set. The sequences of 2500 commands are partitioned into blocks, where each block contains consecutive 100 commands. For each experiment, the input consists of blocks of 100 commands from each user. To draw analogy with the author identification task, each command corresponds to a token and each block of 100 commands represents a sequence.

Parameter Estimation

The RDM algorithm requires a few parameters to be estimated

TABLE I
COMPARISON OF CLASSIFIERS FOR AUTHOR IDENTIFICATION TASK

Classifiers	Accuracy rate (%)
Naïve Bayes	91.5
SVM	98.4
RDM [19]	94
RDM with Naïve Bayes	91.5
RDM with SVM	98.5

for the classification model. The parameters to be selected include 1) size of the window, 2) maximum number of gaps allowed in the window, 3) weights assigned to the classifier at each level, 4) upper and lower cut-offs threshold for key dominant patterns. A greedy search over the parameter space is conducted to determine the best set of parameters. In order to access the current parameter values, the training set is further split into two parts. A classifier is trained on the larger part, and tuned on the smaller part (called the tuning set).

2) Results

We compare four classifiers – Naïve Bayes, RDM using statistical information as features, SVM, and RDM with dominant patterns as features. For SVM, we ignored the most frequent and the least frequent tokens, and used the rest of tokens as features. For our version of RDM, we applied RDM with both Naïve Bayes and SVM in an ensemble of classifiers. We used *SVMLight* as the SVM implementation. The performance of each classifier is presented in Table I. All

classifiers achieve higher than 90% accuracy rate, indicating that this particular classification task is fairly easy. RDM in an ensemble setting does not show any significant improvement over Naïve Bayes and SVM classifiers. On further investigation it was evident that RDM overfits the data, i.e., the training accuracy increases while the testing accuracy decreases. Both SVM and ensemble based RDM with SVM outperform the previous version of RDM. Experiments on the SEA dataset were performed to ensure that RDM is effective for such a setting as compared to the previous RDM version [19]. The superiority of RDM can be seen for more complex tasks, in the following section.

B. Role Identification Task

We show that RDM performs better as compared to Naïve Bayes, Support Vector Machine and Predictive Association Rule based (CPAR [21]) classifiers. CPAR combines the advantages of associative and traditional rule-based classifiers. Support Vector Machines based classifiers have been shown [7] to perform well for text classification tasks. RDM does not use any semantic tools (part-of-speech tagging or synonym

TABLE II
DATASET FOR ROLE IDENTIFICATION

	Training Set	Testing Set	Total	# Sent folders
CEO	1010	250	1260	3
Manager	1403	349	1752	4
Trader	654	162	816	4
VP	1316	327	1643	4
Total	4383	1088	5471	15

groups) in order to extract patterns that later serve as features for the classifiers. As a result, we compare with other techniques that do not utilize domain or semantic knowledge either. A brief introduction to the Enron dataset is provided before the discussion on the experimental setup.

1) Data Preparation and Experimental Setup

Experiments are performed on the March 2, 2004 version of Enron dataset, distributed by William Cohen, <http://www.cs.cmu.edu/~enron/>. The dataset was cleaned to eliminate attachments, quoted text and tables from the body of the email messages and header fields from the email. No effort was made to correct spelling errors or to expand abbreviations in order to reduce noise in the data. Stemming was performed on the dataset.

For the purpose of identifying roles, employees were partitioned into groups based on their organizational role in Enron, as suggested in [22]. Only the roles CEO, Manager, Trader and Vice-president were used in our experiments due to a large number of employees designated with these roles. Since we are concerned with identifying roles based on the sent messages we only deal with messages in the Sent folder of each participant. For each of these roles, the emails are divided into two sets as summarized in Table II. Finally, each stemmed word in an email is considered a token, and each email represents one sequence.

2) Results

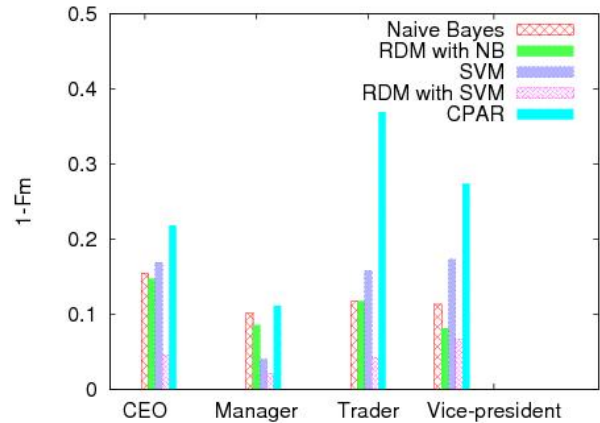


Fig 4. Binary Classification: (1 - F-score)

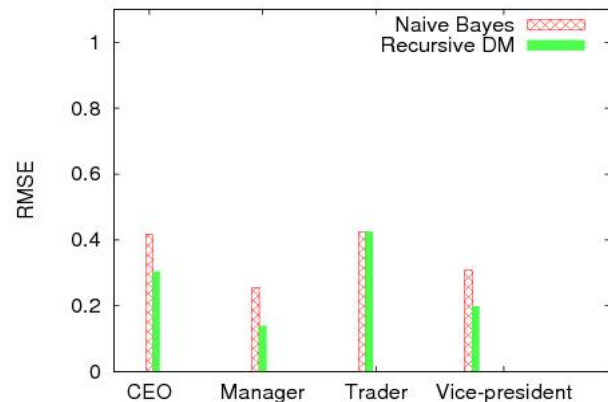


Fig 5. Binary Classification - RMSE Comparison

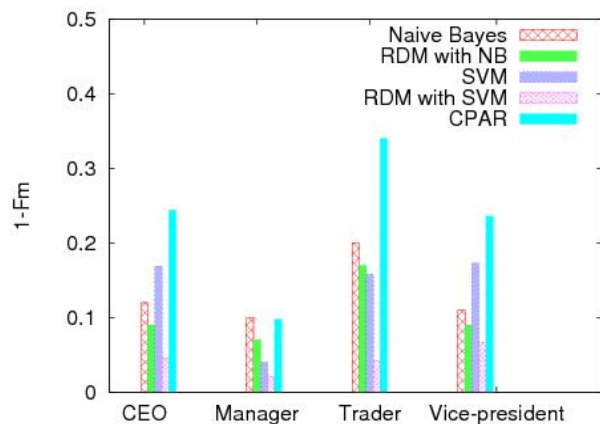


Fig 6. Multi-class Classification: (1 - F-score)

Binary and multi-class classification: We compare the four classifiers – Naïve Bayes, RDM, SVM and CPAR – under two classification settings. RDM was used both with Naïve Bayes and SVM in an ensemble of classifiers. The previous version of RDM [19] was dropped out of this experiment because the learned model always predicted every instance as the major class. In the binary classification setting, given a test message m , the task is to answer “Is message m sent by a person with role r ?” where $r \in R = \{\text{CEO, Manager, Trader, Vice-}$

president}. The training set is divided such that all messages belonging to role r form the positive class and all messages belonging to $R \setminus r$ form the negative class. The performance for the four classifiers is shown in Figure 4, where the values of $(1 - F\text{-measure})$ are presented to highlight the differences in performances. Note that a smaller value of $(1 - F\text{-measure})$ indicates a better classifier. In terms of the F-measure, RDM performs better than NB and SVM. RDM outperforms CPAR under all settings. To further analyze the results, we computed the RMSE (Root Mean Square Error) for NB and RDM. The RMSE is computed using the expression

$$RMSE(SEQ_{test}) = \sqrt{\frac{\sum_{i=1}^{|SEQ_{test}|} (1 - P(r | SEQ_{test}^i))^2}{|SEQ_{test}|}}$$

SEQ_{test}^i is the i^{th} document in the test set and $r = \text{argmax}_c \mathbf{P}(c | SEQ_{test}^i)$. Since the decision function value from *SVMLight* could not be converted to an error term, the plot in Figure 5 does not show comparison with SVM. Similarly, CPAR does not provide any comparable measure. The lower the RMSE value, the more confident the classifier is in its prediction. Figure 5 shows that RDM is more confident in its predictions even when the F-scores for RDM and NB might be very close for a certain role.

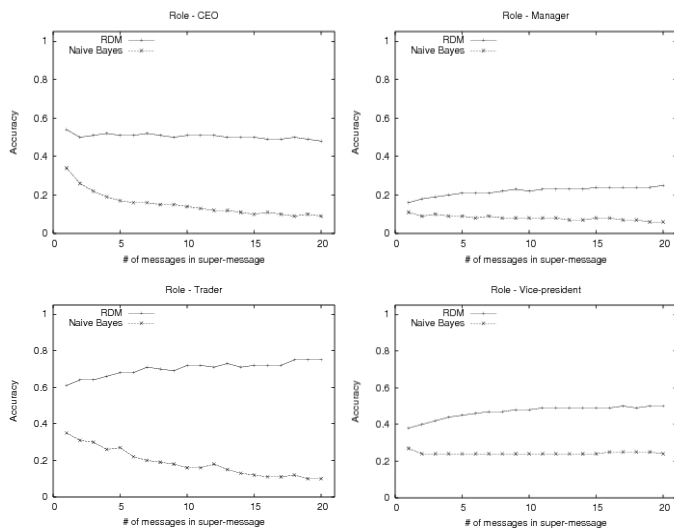


Fig 7. Classification Probability over Unseen Message Folder

The second set of results compares the performance under the multi-class classification setting, wherein the task is to answer “Which is the most likely role, out of roles R_1, \dots, R_n , for sender of message m ?” For NB and RDM, the training data

is split into 4 groups and probabilities computed for each of the roles. For SVM, four sets of datasets are generated, one each for role $(r, R \setminus r)$ pairs. The comparison for the classifiers is shown in Figure 6. RDM outperforms the other classifiers convincingly.

Paired t-test: To further investigate the results obtained for the multi-class scenario, we performed the paired t-test for statistical significance. The paired t-test provides a hypothesis test of the difference between population means for a pair of random samples whose differences are approximately normally distributed. Note that a pair of samples, each of which may not be from a normal distribution, often yields differences that are normally distributed. The null hypothesis H_0 states that the difference in performance between RDM and the other methods is not significant. In other words, H_0 states that the two methods perform equally well. The alternate hypothesis, states otherwise.

A 20-fold cross validation was performed on the data. The accuracy results obtained therein are used for the t-test, where SVM and CPAR are compared against RDM with SVM, and NB is compared against RDM with NB. The results are shown in Table III. Based on the p-value in Table III we reject the null hypothesis, indicating a definite improvement with RDM. The confidence interval for the mean difference shows that the improvement lies between 1.8% and 3% as compared to NB whereas as compared to SVM (and CPAR) it lies between 8% and 10%.

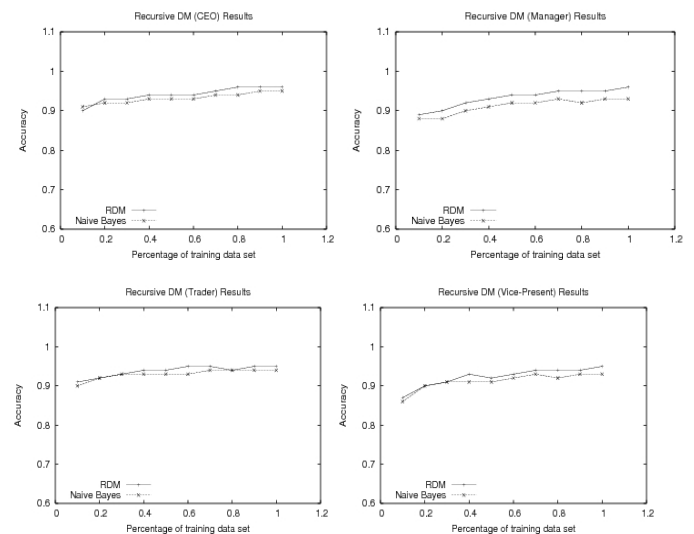


Fig 8. Effect of Changing Training Data Size

Accuracy on unseen user: For the final test we divide each role into two parts based on the users. For instance, the folders of *Jeff Skillings*, *David Delaney* and *John Lavorato* form the CEO group². The first part, namely training set, contains

TABLE III
RESULTS OF PAIRED T-TEST

Classifier Pair	Mean Difference	Std. Dev. of (d)	t-statistic (df=19)	p-value	95% confidence interval
NB vs RDM with NB	0.02393	0.002525	9.48	1.23E-08	(0.0186 – 0.0292)
SVM vs RDM with SVM	0.08927	0.00434	20.55	1.94E-14	(0.0818 – 0.0984)
CPAR vs RDM with SVM	0.09329	0.00535	17.45	3.74E-13	(0.0821 – 0.1045)

¹ A \ B implies all elements of set A after removing elements of set B

² CEOs of Enron subsidiaries are considered as Enron CEOs in this test.

messages from *John Lavorato*, *David Delainey* while messages from *Jeff Skillings* form the second part (test set). An RDM based classifier is trained using messages in the first part and tested on messages in the second part. In this experiment we analyze the performance of the classifier for a member whose messages are not in the training set. The results for different roles are shown in Figure 7. The test set size is gradually increased and the accuracy is noted. Notice that for the roles Manager, Trader and Vice-president the accuracy increases with larger number of message. For the CEO role, there is a marginal decline in the accuracy on increasing the number of test messages. On examining the messages for the CEO, we observed that most of the messages were written by a secretary. This explains the poor performance for the CEO role.

C. Effect of Parameter Changes

In this section, we take a quick look at the effects of varying certain parameters within RDM using the Enron data set. Figure 8, shows the variation in accuracy with increasing training set size. The training set for each of the roles is increased in steps of 10% of the total training set size. From these results we observe that RDM consistently performs as good as or better than NB. Moreover, it shows that both classifiers are quite robust and attain a fairly high accuracy even for smaller training set sizes.

Figure 9a captures the effect of varying window size on overall accuracy of the multi-class setting. The maximum number of gaps is set to 1. The time taken to build the classifier is shown in Figure 9b. From Figure 9a we see that the accuracy is best for a window size of 3 and reduces as the window size is increased. This result is intuitive as larger significant patterns are captured by merging smaller significant patterns, whereas on the other hand smaller patterns cannot be captured using a large window size. Results for the runtime indicate that increasing the window size increases the runtime due to a larger number of generated candidate patterns.

VI. CONCLUSION AND FUTURE WORK

We propose a general framework for feature extraction from a sequence of tokens. The framework is based on the idea of capturing statistically significant sequence patterns at increasing levels of generalization. These patterns act as features for an ensemble of classifiers, one at each level. The proposed method is simple and flexible, such that it can be applied to a range of applications. We applied it for capturing stylistic patterns in the SEA and Enron datasets, which are later used for identifying the authors and the organizational roles of authors respectively.

The method, in its current state, is devoid of any semantic knowledge, which can be easily incorporated to identify semantically related patterns. Techniques such as part of speech tagging and synonym dictionaries can augment our approach. Based on the success of the method on a noisy dataset, the authors believe that the method can perform better on cleaner and structured datasets such as the Reuters dataset

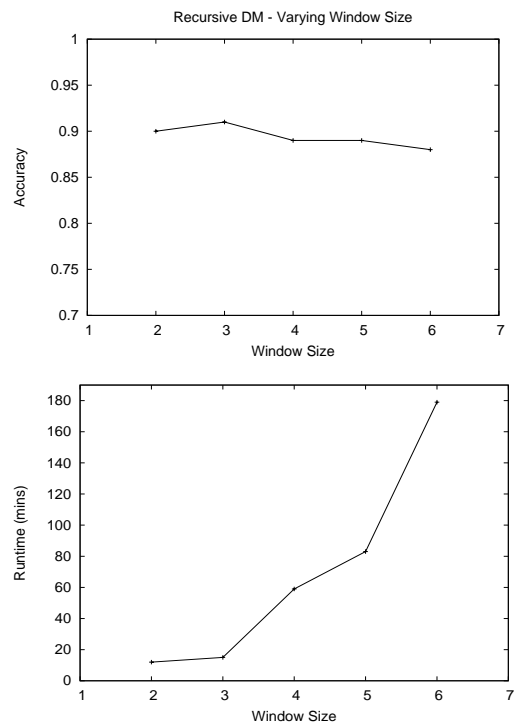


Fig 9. Accuracy (9a) and Runtime (9b) for Varying Window Size

or even on a foreign language dataset, such as Russian Blogosphere data. Moreover, the method can be applied on other application areas such as grouping gene products by their families.

ACKNOWLEDGMENT

We would like to thank the anonymous reviewers for their comments which have helped improve the quality of the paper.

This work was partially supported by the ONR Contract N00014-06-1-0466. The content of this paper does not necessarily reflect the position or policy of the U.S. Government, no official endorsement should be inferred or implied.

REFERENCES

- [1] H. Cheng, X. Yan, J. Han, and C. Hsu, "Discriminative Frequent Pattern Analysis for Effective Classification," *ICDE*, 2007, pp. 716–725.
- [2] S. Coull, J. Branch, B. Szymanski and et al. "Intrusion Detection: A Bioinformatics Approach," *19th Annual Computer Security Applications Conference*, Las Vegas, Nevada, 2003.
- [3] N. Du and B. Wu and X. Pei and B. Wang and L. Xu, "Community detection in large-scale social networks", *WebKDD/SNA-KDD Workshop*, 2007.
- [4] M. Goldberg, M. Hayvanovych, A. Hoonlor, S. Kelley, M. Magdon-Ismael, K. Mertsalov, B. Szymanski and W. Wallace. "Discovery, Analysis and Monitoring of Hidden Social Networks and Their Evolution," *IEEE International Conference on Technologies for Homeland Security*, Waltham, MA, (2008).
- [5] P. Good, *Permutation Tests: A Practical Guide to Resampling Methods for Testing Hypotheses*, Springer, (2000).
- [6] I. Guyon and A. Elisseeff, "An introduction to variable and feature selection," *J. Mach. Learn. Res.*, 3 (2003), pp. 1157–1182.

- [7] T. Joachims, "Text categorization with support vector machines: learning with many relevant features," *Proceedings of 10th ECML*, (1998).
- [8] S. Karlin, and V. Brendel, "Chance and Statistical Significance in Protein and DNA Sequence Analysis," *Science*, 257 (1992), pp. 39–49.
- [9] S. Karlin, "Statistical significance of sequence patterns in proteins," *Current Opinion Struct Biol.*, 5:3 (1995), pp.360–371(12).
- [10] W. Lee, and S. Stolfo, "Data Mining Approaches for Intrusion Detection," *7th USENIX Security Symposium*, (1998), pp. 79–93.
- [11] W. Lee and S. Stolfo, "A Framework for Constructing Features and Models for Intrusion Detection Systems", *ACM Transactions on Information and System Security*, 2000.
- [12] T. Li and M. Ogihara, "Music artist style identification by semi-supervised learning from both lyrics and content," *Proceedings of 12th Annual ACM Multimedia*, 2004, pp. 364–367.
- [13] R. Maxion, "Masquerade Detection using Enriched Command Lines", *International Conference on Dependable Systems and Networks*, 2003.
- [14] C. G. Nevill-Manning and I. H. Witten, "Identifying hierarchical structure in sequences," *Journal of Artificial Intelligence Research*, 7 (1997), pp. 67–82.
- [15] M. Schonlau, W. DuMouchel, and et al. "Computer intrusion: Detecting masqueraders," *Statistical Science*, 16(1), (2001), pp. 58-74.
- [16] J. Seo, and S. Cha, "Masquerade Detection based on SVM and Sequencebased User Commands Profile," *Proceedings of the 2nd ACM symposium on Information, computer and communications security*, Singapore, (2007), pp. 398–400.
- [17] Z. Solan, D. Horn, E. Ruppim and S. Edelman, "Unsupervised learning of natural languages," *Proc Natl Acad Sci U S A*, 102:33 (2005), pp. 11629–11634.
- [18] K. Sripanidkulchai, B. Maggs, and H. Zhang, Efficient content location using interest-based locality in peer-to-peer systems, In *Proc. of IEEE INFOCOM*, (2003).
- [19] B. Szymanski and Y. Zhang, "Recursive Data Mining for Masquerade Detection and Author Identification," *Proc. 5th IEEE SMC IA Workshop*, 2004, pp. 424–431.
- [20] C. Yang and T. Ng, "Terrorism and Crime Related Weblog Social Network: Link, Content Analysis and Information Visualization", *IEEE International Conference on Intelligence and Security Informatics*, 2007.
- [21] X. Yin, and J. Han, "CPAR: Classification based on Predictive Association Rules," *SDM* 2003.
- [22] <http://isi.edu/~adibi/Enron/Enron Employee Status.xls>