International Journal of Hybrid Information Systems, vol. 7(3):89-100, May 2010

Recursive Data Mining for Role Identification in Electronic Communications

Authors:

Vineet Chaoji, Apirak Hoonlor, Boleslaw K. Szymanski

Rensselaer Polytechnic Institute

Troy, New York 12180, USA

{chaojv, hoonla, szymansk}@cs.rpi.edu

Corresponding author:

Boleslaw K. Szymanski

Rensselaer Polytechnic Institute

Troy, New York 12180, USA

Tel: 518-276-2714

FAX: 518-276-4033

Email: szymansk@cs.rpi.edu

Abstract

We present a text mining approach that discovers patterns at varying degrees of abstraction in a hierarchical fashion. The approach allows for certain degree of approximation in matching patterns, which is necessary to capture non-trivial features in realistic datasets. Due to its nature, we call this approach *Recursive Data Mining* (RDM). We demonstrate a novel application of RDM to *role identification* in electronic communications. We use a hybrid approach in which the RDM discovered patterns are used as features to build efficient classifiers.

Since we want to recognize a group of authors communicating in a specific role within an Internet community, the challenge is recognize possibly different roles of an author within different communication communities. Moreover, each individual exchange in electronic communications is typically short, making the standard text mining approaches less efficient than in other applications. An example of such a problem is recognizing roles in a collection of emails from an organization in which middle level managers communicate both with superiors and subordinates. To validate our approach we use the Enron dataset which is such a collection. The results show that a classifier that uses the dominant patterns discovered by Recursive Data Mining performs well in role identification.

Keywords: Data Mining, Feature Extraction or construction, Text classification

1 Introduction

The problem of understanding characteristics of data has attracted keen interest of scientists from early years of computer science. Specifically, characteristics that represent a certain “style” within the data have been widely used to analyze and discriminate between different attributes of data, their sources and the underlying generative models.

Within the data mining community, the term *feature extraction* is commonly used for techniques that identify features relevant to the application at hand. Within this context, the term feature has been loosely used for attributes of data that can range, for instance, from keywords for text documents to principle eigenvectors for high dimensional genetic data. Feature extraction is broadly considered to be composed of two sub-tasks – *feature construction* and *feature selection* [9], each addressing one of the two main challenges of the problem. The first challenge results from the presence of a large amount of noise in the data which results in construction of ineffective features. The second challenge results from the large number of features usually generated. The features are ranked based on optimality criteria – such as information gain, kernel and novelty detection – and only the top-ranked features are used to avoid the curse of dimensionality and enhance generalization capabilities [6].

Every human communication carries not only semantic content defining its meaning but also a unique word and pattern of words structure characteristic of its author. Hence, feature extraction enables reliable detection of a communication’s authorship in the process of *authorship assessment.* In this paper, we present an extension of such a problem to detecting roles of a group of authors. To this end, we apply a general feature extraction method, termed *Recursive Data Mining* (RDM), that uses statistically significant, approximately matched sequential patterns.

Identifying patterns using significance tests was used extensively in biological sequence analysis [13]. In this paper, we focus on extracting patterns from electronic media; those patterns are later used to build a classifier. The approach is independent of any semantic information making it amenable to text documents written in different languages. The method also controls degree of flexibility by allowing inexact matching of patterns. Otherwise, presence of noise (in the form of spelling mistakes, use of abbreviations, etc.) would lead to very few matches.

Even though RDM can be applied to data of any nature (time series data, genome data, etc.), we focus here on text documents, and specifically consider the Enron dataset introduced in [15] as a benchmark for email classification. In our work, the features obtained from Enron database are used to identify the organizational role (e.g., manager, president, secretary, etc.) of the sender of an email. Potential applications of our approach include analysis of groups on the Internet of which structure is not clear or not explicitly defined and, in some cases intentionally obstructed or hidden by the group members. Identifying the leaders and followers in such informal groups is of great value for social sciences, network science and security.

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 a role to an individual. We built a framework for discovering statistically significant sequence patterns from a stream of data. The methods developed can find significant sentences in a stream of text such as email, blog or chat-room session. The methods can also find significant regions in a DNA sequence. Previously, RDM was presented in [7] and [23] for social network analysis and masquerade detection. Some of the key contributions of RDM approach are as follows:

- 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.
- 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. We discuss related work in Section 2. Section 3 introduces the basic terminology for this work and is followed by Section 4 containing a detailed description of our methodology. The experimental results are presented in Section 5, while Section 6 offers conclusions.

2 Related Work

There is a large body of work that deals with extracting patterns and features from unstructured raw text data. Experts in the fields of information retrieval, natural language processing, data mining and statistical learning have focussed on a diverse set of techniques for feature extractions and feature selections to solve the author identification problem. Linguists use statistical techniques to obtain a set of significant words that would help identify authors. In 1964 Mosteller and Wallace [16] solved the Federalist Papers problem by identifying 70 *function words* and applying statistical inference for the analysis.

In the Natural Language Processing (NLP) community, the Hidden Markov Model has greatly influenced many techniques used for speech recognition [3]. Other complementary techniques used by the NLP community include part of speech tagging [10], which is used for assigning a syntactic category to each word in a text document. A comprehensive introduction to feature selection techniques is presented in [9]. Many other applications also benefit from feature extraction and feature selection, which include automatic painting classification [5], detection of malicious alterations of images [20], grouping proteins and genes into classes based on function, location and biological process in bioinformatics [25] and characterizing the behavior of an individual or a group of individuals in e-Commerce [24].

The task of role identification is a special form of document classification, also referred to as text categorization. Sebastiani provides a good survey of machine learning applied to text categorization [21]. In [18], Peng and Schuurmans propose the use of n-grams with Naïve Bayes classifier to improve the performance. Our work shares with [17] and [22] an assumption that the underlying structure in English text can be learned in a hierarchical manner. In [17] the authors extract a hierarchical nested structure by substituting grammar for repeated occurrences of segments of tokens. Similarly, in [22] 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 [22]. In recent works [2] and [27], the frequent mining was modified to obtain useful patterns which are used for classification in various domains.

For pattern mining, various significance tests for sequence patterns have been proposed. Permutation test [8] provides 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. ([13] and [14]) have proposed many significance tests for identifying relevant regions in protein sequences. Their approach relies of 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.

3 Preliminaries

Consider a set of sequences, denoted as $\mathcal{SEQ}$. Each sequence consists of a series of *tokens* from a set $\mathcal{T}$. Thus, a sequence $S \in \mathcal{SEQ}$ of length n can be represented as $t_1, t_2, \ldots, t_n$, where $t_i \in \mathcal{T}$. Depending on the application, a token may represent a different entity. For instance, in the domain of 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 numeric value (price and volume) while the sequence would represent 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 $\{s_i : s_1, s_{l(\mathcal{P})} \in \mathcal{T} \wedge s_j \in \mathcal{T} \cup \{\perp\}, j = 2 \ldots l(\mathcal{P}) - 1\}$, where i is the index of a token in the sequence and $l(\mathcal{P})$ is the *length* of the pattern $\mathcal{P}$. It should be noted that the first and last tokens are never the gap token. This restriction is useful for combining contiguous patterns.

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 between 0 and 1 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 *s*et 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 $\cup_{g=0}^{max\_gap} S_i(l, g)$ 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. 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.

4 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 accepts the input sequence as the initial sequence for the first iteration in the iterative step of RDM. Note that, a user can apply preprocessing methods, such as stop words and stemming, on the input sequence prior to applying RDM. In the first iteration, 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. Next, the algorithm 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 name 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 is a smaller set of valid matches ($\mathcal{M}$). At higher levels, the patterns are more general, resulting is a larger $\mathcal{M}$ set. On the other hand, with increasing levels, the number of patterns found decreases 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*, and follows by pattern selection through the *pattern significance assessment* 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 initial sequence of tokens, (*level(0) sequences*), will be referred to as $\mathcal{SEQ}_0$. The set of sequences for level $(i + l)$ 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 $\mathcal{L}$.

4.1 Pattern Generation

A sliding window of length l_w moves over $\mathcal{SEQ}_v$ ($v = 0$ 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. A bounded hash keeps count of the number of occurrences of each pattern at level v, as the sliding window moves over $\mathcal{SEQ}_v$. This forms the first pass over sequence $\mathcal{SEQ}_v$. Figure 1 shows the patterns generated at position 1 and 2 of the sequence.

4.2 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. Recall that the set of unique tokens appearing in a set of sequences $\mathcal{SEQ}$ is denoted by $\mathcal{T}$. The frequency of a token t_i appearing in $\mathcal{SEQ}$ will be denoted by f_{t_i}. So the probability of token t_i over $\mathcal{SEQ}$ is $P(t_i)$, where

$$P(t_i) = \frac{f_{t_i}}{\sum_{j=1}^{|T|} f_{t_j}} \tag{1}$$

For a pattern $\mathcal{P}$ of length l_w, the probabilities of tokens appearing in the pattern can be represented as a vector $(p_{t_1}, p_{t_2}, \cdots, p_{t_{l_w}})$. Recall that a gap is represented by a special token $\perp$. The probability of pattern

$\mathcal{P}$ is thus given by the expression

$$\begin{aligned}\mathbf{P}(\mathcal{P}) &= P(RV_1 = t_1, RV_2 = t_2, \cdots, RV_{l_w} = t_{l_w}) \\ &= p(t_1)p(t_2 \mid t_1) \cdots p(t_{l_w} \mid t_1, \cdots t_{l_w - 1})\end{aligned} \tag{2}$$

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

$$\mathbf{P}(\mathcal{P}) = \prod_{i=1}^{l_w} p_{t_i} \tag{3}$$

The probability of a gap token, denoted as ϵ, is a user defined constant (see 4.3 for details). The probability of occurrence of $\mathcal{P}$ under the independent appearance assumption (random model) is given by

$$\mathbf{P}_R(\mathcal{P}) = P(RV_1 = t_1, RV_2 = t_2, \cdots, RV_{l_w} = t_{l_w}) \tag{4}$$

Since under the random model each token is equally likely to appear, the above expression simplifies to

$$\mathbf{P}_R(\mathcal{P}) = \Big(\frac{1}{\mid \mathcal{T} \mid}\Big)^{l_w}. \tag{5}$$

The ratio $\frac{\mathbf{P}_R(\mathcal{P})}{\mathbf{P}(\mathcal{P})}$ is used to determine significance of the pattern. If the above ratio is smaller than 1, then the pattern is considered significant, otherwise it is considered insignificant. The ratio indicates the likelihood of pattern occurrence 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

$$\mathbf{LRT} = -2log_e\Big(\frac{\mathcal{L}_R(\theta)}{\mathcal{L}_O(\theta)}\Big) \tag{6}$$

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 $\frac{\mathbf{P}_R(\mathcal{P})}{\mathbf{P}(\mathcal{P})}$ smaller than 1 or not. If not, then we can conclude that no pattern generated from this window is significant.

4.3 Dominant Patterns

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 can be computed as follows:

$$score(\mathcal{P}[i], S_v[j]) = \begin{cases} 1 & \text{if } \mathcal{P}[i] = S_v[j] \\ \epsilon & \text{if } \mathcal{P}[i] = \perp, \epsilon < 1 \\ 0 & \text{otherwise} \end{cases} \tag{7}$$

where $\mathcal{P}[i]$ is the i^{th} token of the pattern and j is the corresponding 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 user defined, which is set to be 0.95 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 by

$$score(\mathcal{P}, S_v[j]) = \sum_{i=1}^{|\mathcal{P}|} score(\mathcal{P}[i], S_v[j+i]). \tag{8}$$

The pattern 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 $\operatorname{argmax}_{x \in S_v} score(x, S_v[j])$. The term dominant pattern reflects 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 of the sequence at level v, the sequence for level $v+1$ is generated. 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*. 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. k is selected using the tuning dataset (see Section 5). Figure 3 illustrates the process of collapsing noise tokens into a single special token N. Once the noise tokens are collapsed, distant tokens can now fall within the same window, leading to more patterns being discovered at higher levels. The set of dominant patterns 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.

4.4 Training and Testing Phases

The training phase involves using dominant patterns generated at each level to construct an ensemble of classifiers ($\mathcal{C}_1, \mathcal{C}_2, \cdots, \mathcal{C}_{max\_level}$), one for each level. The dominant patterns reflect the most relevant patterns, ignoring the highly frequent and infrequent patterns (upper and lower cut–offs in the pattern frequency distribution). The upper and lower cut–offs are intended to prevent the use of insignificant patterns as features. The classifiers can be created using any machine learning method, such as Naïve Bayes or Support Vector Machine. Given a set of text documents $\mathcal{SEQ}_{tr}$, along with the labels $r_1, r_2, \cdots, r_v$ of all possible classes, dominant patterns are generated for each document starting at level 0 up to level *max_level*. The union of all tokens in $\mathcal{T}$ and dominant patterns at a level v across all documents in $\mathcal{SEQ}_{tr}$ forms the set of feature for classifier $\mathcal{C}_v$. For the ensemble of classifiers, the final prediction value is the weighted sum of the class prediction of individual classifier. Each classifier is assigned a weight that reflects the confidence of the classifier. There are many weighting schemes for ensemble which can be applied to determine this confidence value (see [1] for more details). For our work, to determine this confidence value, the set $\mathcal{SEQ}_{tr}$ is further split into a training set $\mathcal{SEQ}_{new}$ and a tuning set. Each classifier in the ensemble trains its model based on $\mathcal{SEQ}_{new}$. The accuracy of the classifier on the tuning set determines the confidence of classifier $\mathcal{C}_i$ as

$$conf(\mathcal{C}_i) = \frac{accuracy(\mathcal{C}_i)}{\sum_{j=1}^{max\_levels} accuracy(\mathcal{C}_j)}. \tag{9}$$

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 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. It should be noted that if the best match has a score below the 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 prediction value is assigned based on the following weighting scheme:

$$\mathbf{P}(C \mid x) = \sum_{i=1}^{max\_levels} conf(\mathcal{C}_i) \times \mathbf{P}_{\mathcal{C}_i}(C \mid x) \tag{10}$$

where x is a test sequence and $\mathbf{P}_{\mathcal{C}_i}(C \mid x)$ is the prediction value assigned by classifier $\mathcal{C}_i$.

5 Experiments and Results

There are two sets of experiments presented in section 5.2 and section 5.3, respectively. In the first set of experiments, we use RDM to extract the pattern of ordered words for the role identification task. We show that classifiers based on RDM perform better than comparable classifiers such as Naïve Bayes (NB), Support Vector Machines (SVM) and Predictive Association Rule based (CPAR [26], which, authors claim, combines the advantages of associative and traditional rule-based classifiers). Support Vector Machines based classifiers have been shown by [11] to perform well for text classification tasks. We used SVMLight as the SVM implementation [12], and IlliMine package for CPAR [26]. RDM does not require any semantic tools (part-of-speech tagging or synonym groups) in order to extract patterns that later serve as features for the classifiers. As a result, we compare RDM with other techniques that do not utilize domain or semantic knowledge either. The second set of experiments studies the effects of the training set sizes and the influence of the sliding window size on the performance of RDM on role identification tasks. We focus our attention to RDM with NB and use NB as a base line comparison. A brief introduction to the Enron dataset used for running the experiments is provided before the discussion on the experimental setup.

5.1 Data Preparation and Experimental Setup

Experiments were performed on the March 2, 2004 version of Enron dataset, distributed by William Cohen [4]. 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 an attempt to reduce the noise in the data. We applied Porter stemming from the Snowball Project [19], on the input text documents because it improves the over all performance of all classifiers on the tuning dataset.

For our purpose of identifying roles, employees were partitioned into groups based on their organizational role in Enron, as suggested in [28]. Only the roles *CEO, Manager, Trader and Vice-president* were used in our experiments because a large number of employees were designated with these roles. Since we are concerned with identifying roles based on messages sent by employees, we only deal with the messages in the *Sent* folder of each participant. For each of the roles, the emails are divided into two sets as summarized in Table 1. Finally, each word in an email is considered a token, and each email represents one sequence.

The RDM algorithm requires a few parameters to be set for the classification model. They include 1) the size of the window, 2) the maximum number of gaps allowed in the window, 3) the weights assigned to the classifier at each level, 4) the parameter k used to eliminate noisy tokens. A greedy search over the parameter space is conducted to determine the best set of parameter values. To compute the 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).

5.2 Performance of RDM

We compare five classifiers – Naïve Bayes, RDM with NB, SVM, RDM with SVM and CPAR – under two classification settings: binary and multi-class. RDM was used with both Naïve Bayes, and SVM as the ensemble classifiers. For both classification settings, *F-measure*, also called F-score, [1] is used to compare performance of the classifiers.

In the binary classification setting, given a test message m, the task is to answer the question *"Is message* m *sent by a person with role r"?* where $r \in \mathcal{R} = \{$CEO, Manager, Trader, Vice-president$\}$. The training set is divided in such a way that all messages belonging to role r form the positive class and all messages belonging to $\mathcal{R}\backslash r$ [2] form the negative class. The performance for the five classifiers is shown in Figure 4, where the values of 1 - *F-measure* are presented to highlight the differences in performances. Note that a smaller value of 1 - *F-measure* indicates a better classifier. In terms of the F-measure, RDM with SVM performs better than NB, SVM or CPAR for all tested roles while RDM with NB performs better for most of the roles. To further analyze the results, we computed the Root Mean Square Error (RMSE) for NB and

[1] *F-measure* is the harmonic mean of precision and recall.

[2] $\mathcal{A}\backslash B$ denotes the set difference operation $A - B$ (A minus B).

RDM with NB. The RMSE is computed using the expression

$$RMSE(\mathcal{T}_{test}) = \sqrt{\frac{\sum_{i=1}^{|\mathcal{T}_{test}|}(1 - P(r \mid {\mathcal{T}_{test}}^{i}))^2}{\mid \mathcal{T}_{test} \mid}} \tag{11}$$

where ${\mathcal{T}_{test}}^{i}$ is the i^{th} document in the test set and $r = \text{argmax}_c\, P(c \mid {\mathcal{T}_{test}}^{i})$. 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 with NB is more confident in its predictions even when the F-measure's for RDM with NB and NB might be very close for a certain role.

The second set of results compares the performance under the multi-class classification setting, wherein the task is to answer the question *"Which is the most likely role, out of roles $R_1, \ldots, R_n$, for sender of message* m*?"* For NB and RDM, the training data is split into four groups and probabilities computed for each of the roles. For SVM, four sets of datasets are generated, one each for role (r, $\mathcal{R}\backslash r$) pairs. The comparison for the classifiers is shown in Figure 6. RDM convincingly outperforms the other classifiers.

To further investigate the results obtained for the multi-class scenario, we performed the *paired t-test* for statistical significance. 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 (denoted as RDM-SVM), and NB is compared against RDM with NB (denoted as RDM-NB). The results are shown in Table 2. Based on the *p*-value in Table 2 we reject the null hypothesis, indicating a definite improvement provided by RDM. The confidence interval for the mean difference shows that the improvement lies between 1.8% and 3% for RDM-NB compared to NB alone, whereas RDM-SVM when compared to SVM (and CPAR) provides the improvement between 8% and 10%.

For the final test we divide each role into two parts based on the users. For instance, the folders of *Jeff Skillings, David Delainey* and *John Lavorato* form the CEO group[3]. The first part, namely training set, contains 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. The opposite effect is observed for the role of CEO. On examining the messages for the CEO, we observed that most of the messages were written by secretaries. This explains the poor performance of classifiers for this role.

[3] It should be noted that a CEO of Enron subsidiaries is also considered as an Enron CEO for our experiments.

5.3 Effect of Parameter Changes

In this section, we take a quick look at the effects of varying certain parameters within RDM on the role identification tasks. For this section, we use accuracy for evaluation purposes. Figure 8, shows the variation in accuracy of RDM with NB on the increasing training set size in the binary setting of role identification task. 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 with NB consistently performs as good 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 9, captures the effect of varying window size on overall accuracy of RDM with NB in the multi-class setting of role identification task. The maximum number of gaps is set to 1. Figure 9 shows 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.

6 Conclusion

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, hence, it can be applied to a range of applications. We applied it to capturing stylistic patterns in the Enron dataset and used those patterns for identifying the organizational roles of authors. 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, we believe that the method can perform better on cleaner datasets and on other application areas such as grouping gene products by their families. For our future work, we plan to conduct experiment to demonstrate the broad applicability of this method on gene datasets such as GenBank database. We also plan to apply RDM on text categorization task of short and sparse text data set and a foreign language dataset such as the Russian Blogosphere data, Twitter data and short movie reviews.

Acknowledgment

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] F. C. Bernardini, M. C. Monard and R. C. Prati, Constructing Ensembles of Symbolic Classifiers, International Journal of Hybrid Intelligent Systems, **3** (2006), pp. 159–167

[2] H. Cheng, X. Yan, J. Han, and C. Hsu, Discriminative Frequent Pattern Analysis for Effective Classification, In: *Proc. ICDE*, 2007, pp. 716–725.

[3] K. Church, A stochastic parts program and noun phrase parser for unrestricted text, In: *Proc. 2nd Conference on Applied Natural Language Processing*, February 09-12, 1988, Austin, Texas

[4] W. W. Cohen, Enron Email Dataset. *http://www.cs.cmu.edu/˜enron/*, Last access: May 25th, 2008.

[5] A. Ioana Deac, J. C. A. van der Lubbe and E. Backer, Feature Selection for Paintings Classification by Optimal Tree Pruning, *MRCS*, **4105** (2006), pp. 354–361.

[6] P. F. Evangelista, M. J. Embrechts, and B. K. Szymanski, Taming the Curse of Dimensionality in Kernels and Novelty Detection, *Applied Soft Computing Technologies: The Challenge of Complexity*, A. Abraham, B. Baets, M. Koppen, and B. Nickolay (Eds.), Springer Verlag, Berlin, 2006.

[7] M. Goldberg, M. Hayvanovich, A. Hoonlor, S. Kelley, M. Magdon-Ismail, K. Mertsalov, B. Szymanski, and W. Wallace, Discovery, Analysis and Monitoring of Hidden Social Networks and Their Evolution, In: *Proc. IEEE International Conference on Technologies for Homeland Security*, Waltham, MA, 2008.

[8] P. Good, *Permutation Tests: A Practical Guide to Resampling Methods for Testing Hypotheses*, Springer, 2000.

[9] I. Guyon and A. Elisseeff, An introduction to variable and feature selection, *J. Mach. Learn. Res.*, **3** (2003), pp. 1157–1182.

[10] H. Halteren, J. Zavrel, and W. Daelemans, Improving Accuracy in NLP Through Combination of Machine Learning Systems, *Computational Linguistics.* **27(2)** (2001), pp. 199-229.

[11] T. Joachims, Text categorization with support vector machines: learning with many relevant features, In: *Proc. 10th ECML*, 1998.

[12] T. Joachims, Making large-Scale SVM Learning Practical, *Advances in Kernel Methods - Support Vector Learning*, B. Schlkopf, C. Burges and A. Smola (ed.), MIT-Press, 1999.

[13] S. Karlin, and V. Brendel, Chance and Statistical Significance in Protein and DNA Sequence Analysis, *Science*, **257** (1992), pp. 39–49.

[14] S. Karlin, Statistical significance of sequence patterns in proteins, *Current Opinion Struct Biol.*, **5(3)** (1995), pp.360–371(12).

[15] B. Klimt and Y. Yang, The Enron Corpus: A New Dataset for Email Classification Research, In: *Proc. ECML*, 2004, pp. 217–226.

[16] F. Mosteller, and D. L. Wallace, *Inference and disputed authorship: The Federalist*, Reading, Mass., Addison-Wesley, 1964.

[17] C. G. Nevill-Manning and I. H. Witten, Identifying hierarchical structure in sequences, *Journal of Artificial Intelligence Research*, **7** (1997), pp. 67–82.

[18] F. Peng and D. Schuurmans, Combining Naive Bayes and n-Gram Language Models for Text Classification, *ECIR*, 2003.

[19] M. F. Porter, *The Porter Stemming Algorithm*, http://snowball.tartarus.org/index.php, January, 2006.

[20] C. Rey and J. Dugelay, Blind detection of malicious alterations on still images using robust watermarks, In: *Proc. IEEE Seminar Secure Images and Image Authentication*, 2000, pp. 7/1–7/6.

[21] F. Sebastiani, Machine learning in automated text categorization, *ACM Computing Surveys*, **34(1)** (2002), pp. 1–47.

[22] Z. Solan, D. Horn, E. Ruppin and S. Edelman, Unsupervised learning of natural languages, In: *Proc Natl Acad Sci U S A*, **102(33)** (2005), pp. 11629–11634.

[23] B. Szymanski and Y. Zhang, Recursive Data Mining for Masquerade Detection and Author Identification, In: *Proc. 5th IEEE SMC IA Workshop*, 2004, pp. 424–431.

[24] H. Yang, Z. Pan, X. Wang, and B. Xu, A personalized products selection assistance based on e-commerce machine learning,*ICMLC*, **4** (2004), pp. 2629–2633.

[25] M. Yousef, S. Jung, L. C. Showe, and M. K. Showe, Recursive Cluster Elimination (RCE) for Classification and Feature Selection from Gene Expression Data, *BMC Bioinformatics*, **8** (2007).

[26] X. Yin, and J. Han, CPAR: Classification based on Predictive Association Rules, In: *Proc. SDM*, 2003.

[27] M. Zaki, C. Carothers, and B. K. Szymanski, VOGUE: A Novel Variable Order Hidden Markov Model using Frequent Sequence Mining, *ACM Transactions on Knowledge Discovery from Data*, to appear, 2009.

[28] Enron Employee Status Record. *http://isi.edu/˜adibi/Enron/Enron Employee Status.xls*, Last access: July 10th, 2007.

Algorithm 1 Outline of Recursive Data Mining Algorithm

```
Input: Set of sequences $\mathcal{SEQ}_0$
Output: Sets of patterns (features) $\mathcal{L}$, one for each level
$\mathcal{L} = \{\}$, $i = 0$
repeat
    if $i > 0$ then
        $\mathcal{SEQ}_i = make\_next\_level(\mathcal{SEQ}_{i-1}, \mathcal{D})$ // Level(i)
    end
    $\mathcal{P}_{ALL} = pattern\_generation(\mathcal{SEQ}_i)$
    $\mathcal{P}_{SIG} = sig\_patterns(\mathcal{SEQ}_i, \mathcal{P}_{ALL})$
    $\mathcal{D} = get\_domi\_patterns(\mathcal{SEQ}_i, \mathcal{P}_{SIG})$
    $\mathcal{L} = \mathcal{L} \cup \mathcal{D}$
    $i{+}{+}$
until $\mathcal{D} == \varnothing \vee i == max\_level$
return $\mathcal{L}$
```

Table 1: Dataset for Role Identification task.

ROLE	Training Set Size	Testing Set Size	Total Size	# 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

Table 2: Results of paired t-test.

Classifier Pair	Mean difference	Standard Deviation of $(\overline{d})$	t-statistic (df=19)	p-value	95% confidence interval
NB vs RDM-NB	0.02393	0.002525	9.48	1.23E-08	(0.0186 - 0.0292)
SVM vs RDM-SVM	0.08927	0.00434	20.55	1.94E-14	(0.0818 - 0.0984)
CPAR vs RDM-SVM	0.09329	0.00535	17.45	3.74E-13	(0.0821 - 0.1045)

Figure captions

Fig. 1: Pattern generation step. The left most column shows a subset of generated patterns from the first position in the sequence with window size $l_w = 6$ and maximum number of gap allowed $max\_gap = 2$. The right most column shows a subset of generated patterns from the second position in the same sequence under the same setting.

Fig. 2: Sequence re-writing step. From the level-0 sequence, three subsequence matches two of the dominant patterns (displayed in the table to the right). The level-1 sequence is the result of replacing dominant patterns 80 and 81 to their corresponding matches in level-0 sequence

Fig. 3: Removing noisy tokens for long range patterns. The token that is considered as a noisy token is replaced by symbol "N" during the rewriting step. If a concatenated sequence of N tokens appears after symbol replacement, it is collapsed into a single symbol of N.

Fig. 4: Results of binary classification setting on role identification tasks – y-axis indicates (1-F-measure).

Fig. 5: RMSE comparison results of binary classification setting on role identification tasks.

Fig. 6: Results of multi-class classification setting on role identification tasks – y-axis indicates (1-F-measure).

Fig. 7: Classification probability over unseen message folder. Experiments are performed on binary classification setting over four roles: CEO, Manager, Trader and Vice-president.

Fig. 8: Effect of changing training data size on RDM with NB and NB. Experiments are performed on the binary classification setting over four roles: CEO, Manager, Trader and Vice-president

Fig. 9: Accuracy of RDM with NB on varying window size. Experiments are performed on the multi-class setting of role identification task.

Fig. 1

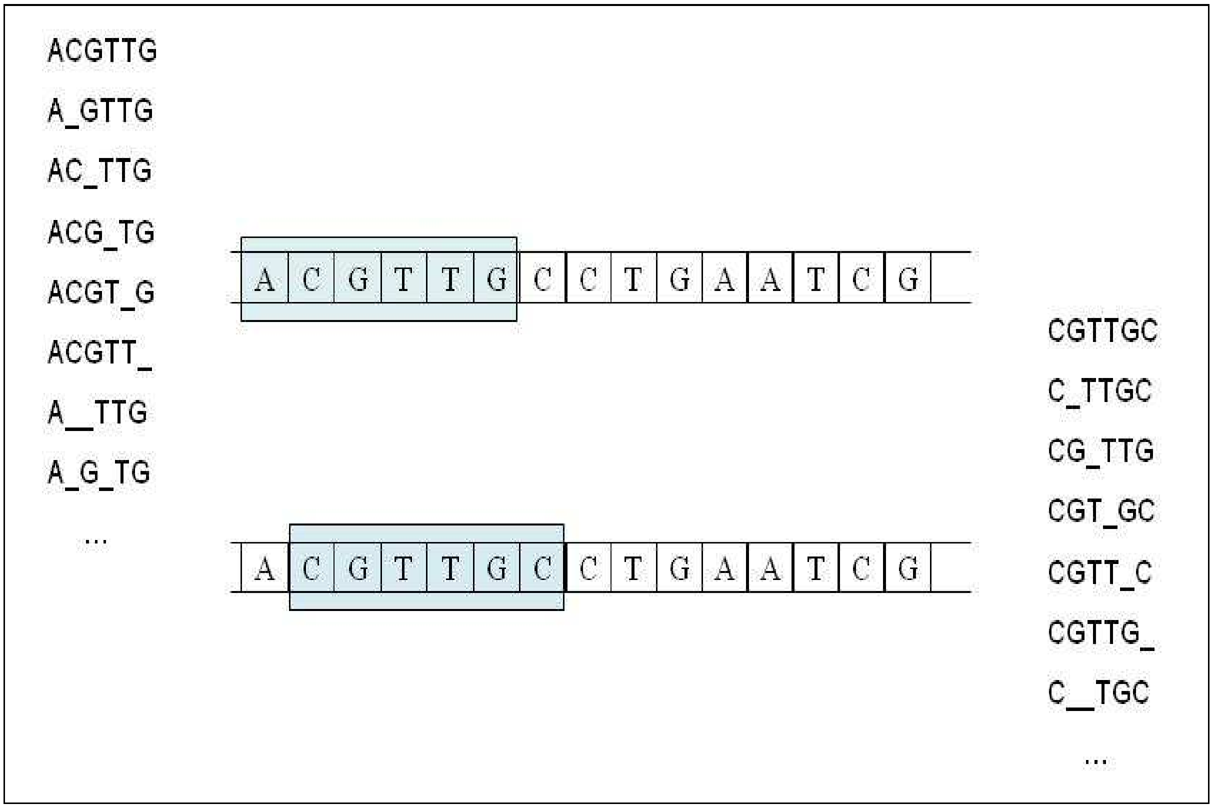
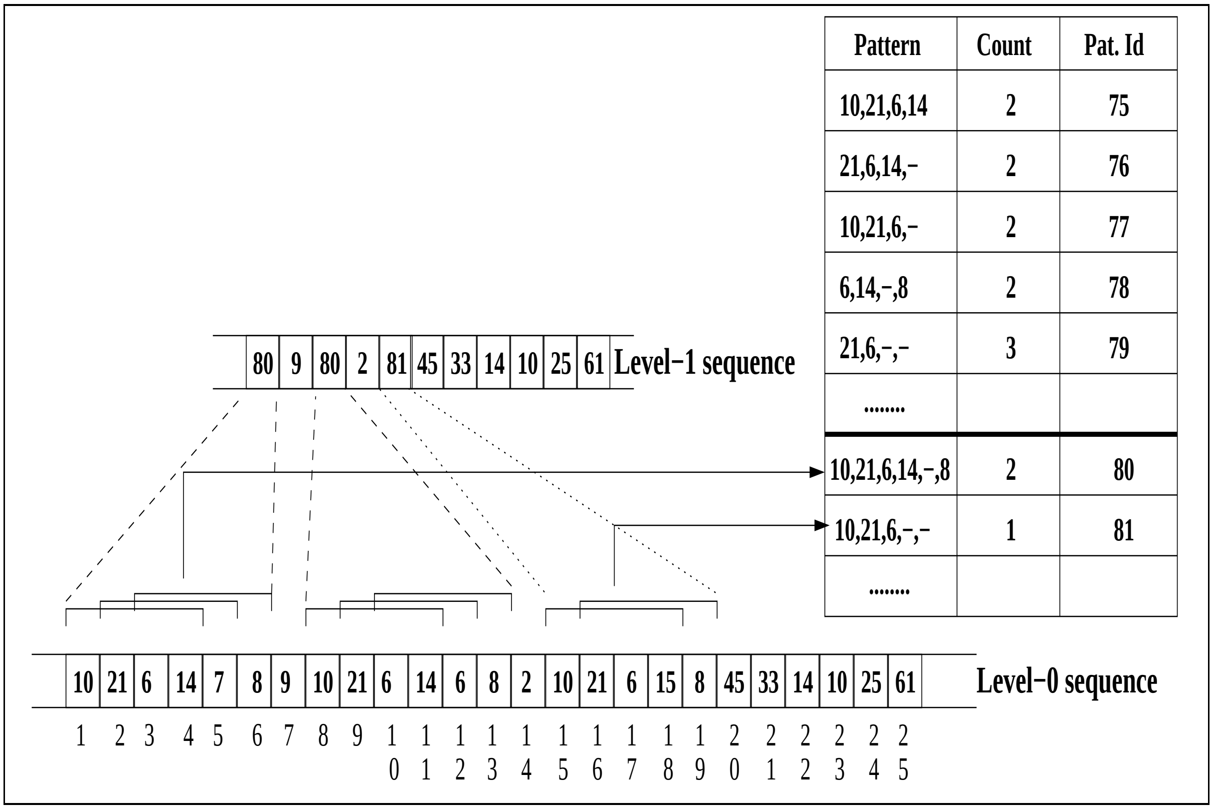
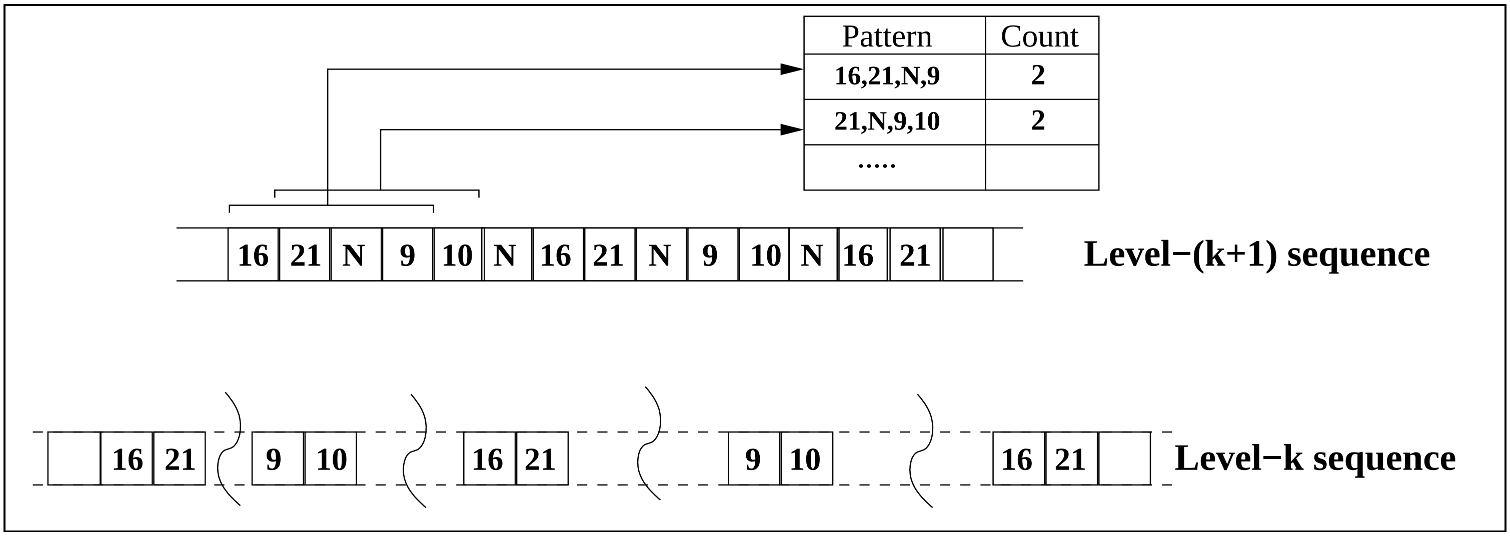

ACGTTG
A_GTTG
AC_TTG
ACG_TG
ACGT_G
ACGTT_
A__TTG
A_G_TG
...
A C G T T G C C T G A A T C G
A C G T T G C C T G A A T C G
CGTTGC
C_TTGC
CG_TTG
CGT_GC
CGTT_C
CGTTG_
C__TGC
...

Fig. 2



Pattern	Count	Pat. Id
10,21,6,14	2	75
21,6,14,−	2	76
10,21,6,−	2	77
6,14,−,8	2	78
21,6,−,−	3	79
....		
10,21,6,14,−,8	2	80
10,21,6,−,−	1	81
....		

Fig. 3



Pattern	Count
16,21,N,9	2
21,N,9,10	2
.....	

Fig. 4

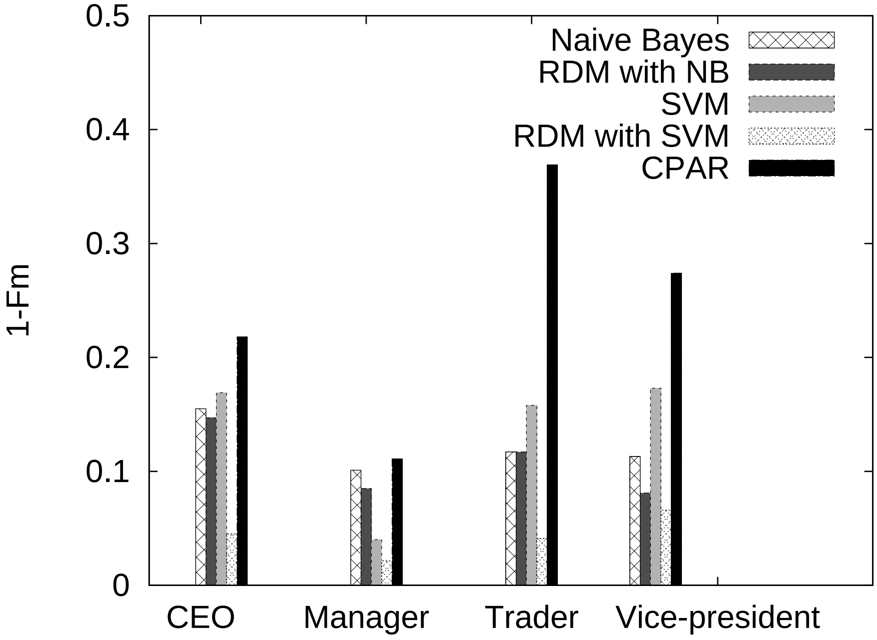

0.5
0.4
0.3
0.2
0.1
0
1-Fm
Naive Bayes
RDM with NB
SVM
RDM with SVM
CPAR
CEO
Manager
Trader
Vice-president

Fig. 5

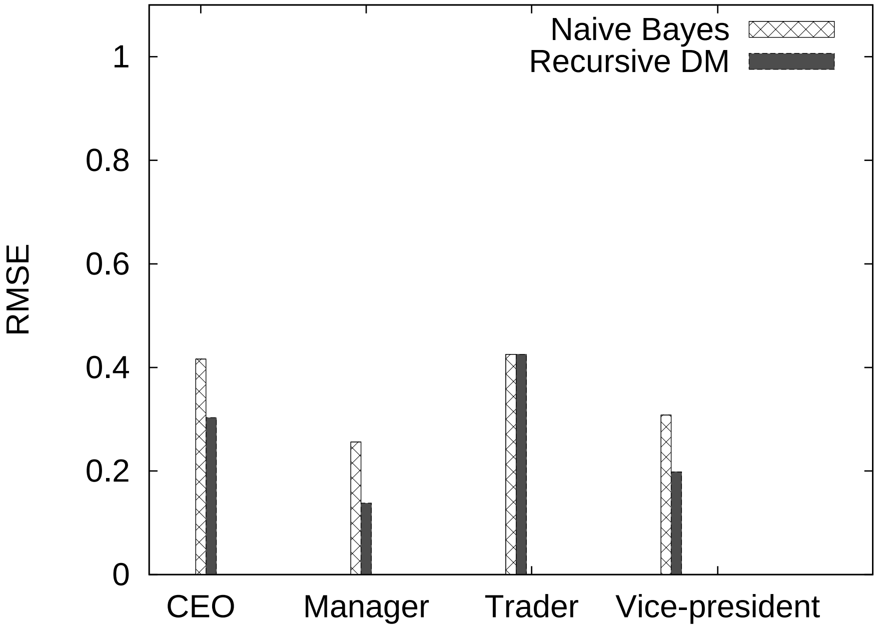
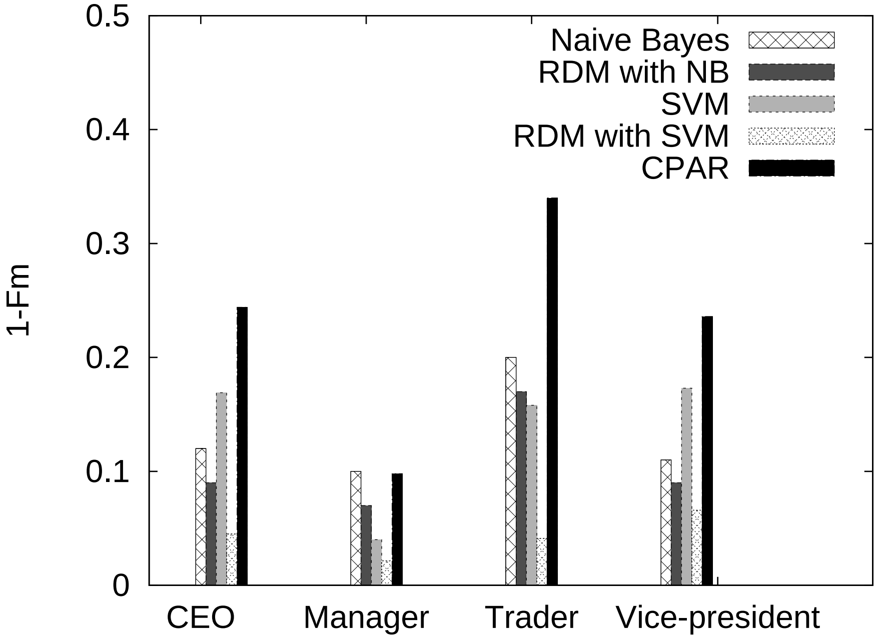

Naive Bayes
Recursive DM
RMSE
1
0.8
0.6
0.4
0.2
0
CEO
Manager
Trader
Vice-president

Fig. 6



Naive Bayes
RDM with NB
SVM
RDM with SVM
CPAR
1-Fm
0.5
0.4
0.3
0.2
0.1
0
CEO
Manager
Trader
Vice-president

Fig. 7

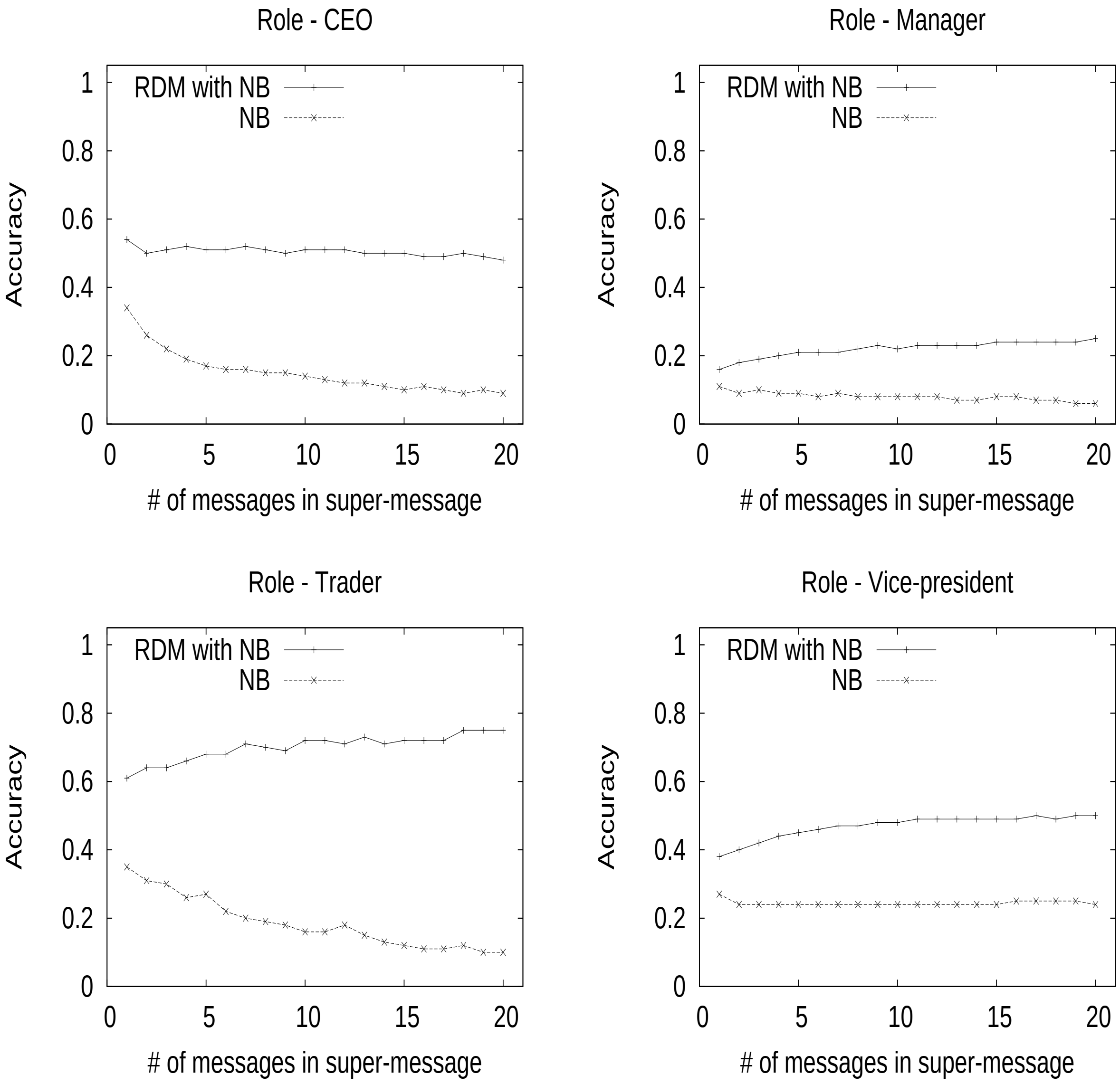

Role - CEO
Role - Manager
Role - Trader
Role - Vice-president
Accuracy
# of messages in super-message
RDM with NB
NB
0
0.2
0.4
0.6
0.8
1
0
5
10
15
20

Fig. 8

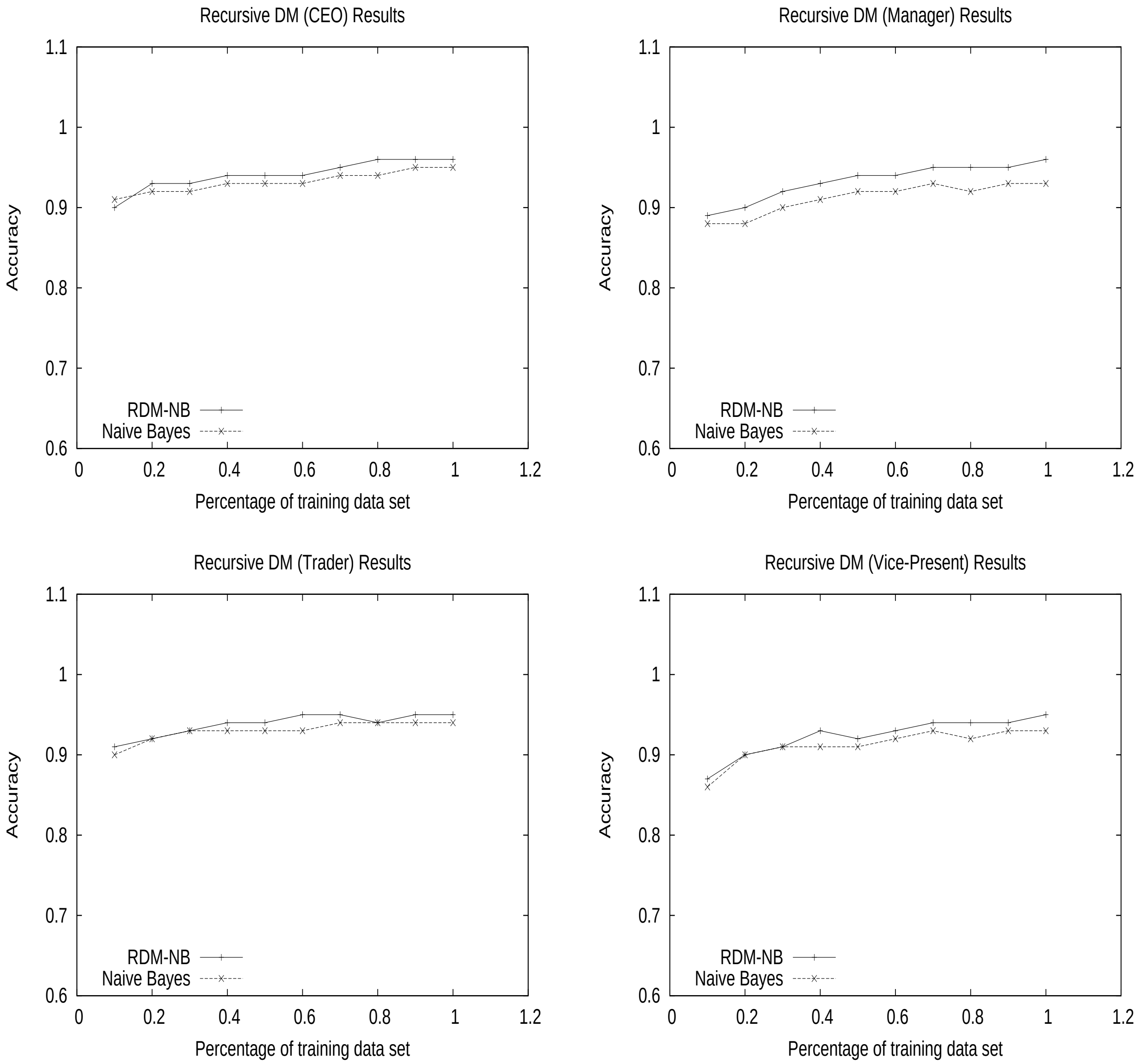
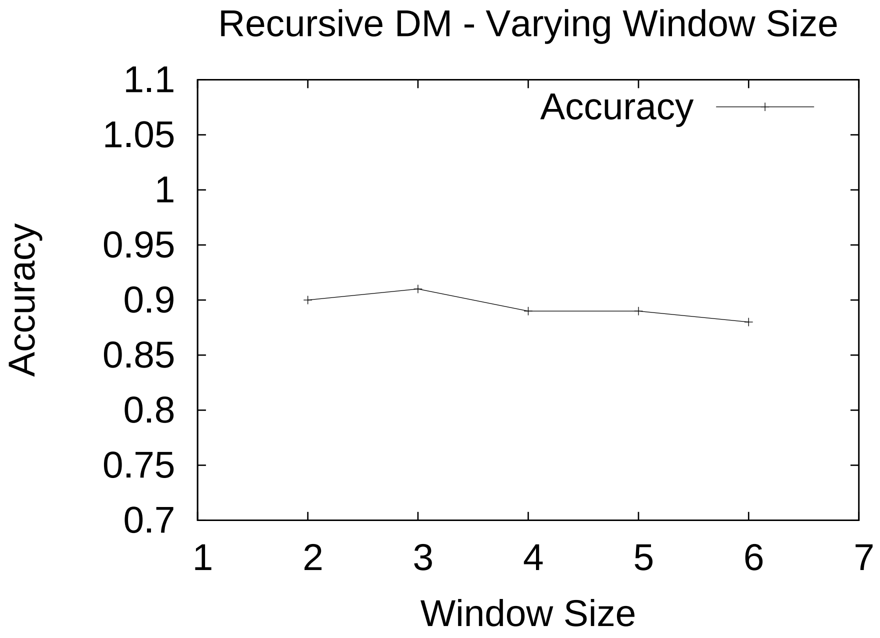

Recursive DM (CEO) Results
Recursive DM (Manager) Results
Recursive DM (Trader) Results
Recursive DM (Vice-Present) Results
Accuracy
Percentage of training data set
RDM-NB
Naive Bayes
1.1
1
0.9
0.8
0.7
0.6
0
0.2
0.4
0.6
0.8
1
1.2

Fig. 9


Recursive DM - Varying Window Size
Accuracy
Accuracy
1.1
1.05
1
0.95
0.9
0.85
0.8
0.75
0.7
1
2
3
4
5
6
7
Window Size