



Department of Computer Science and Engineering
University of Texas at Arlington
Arlington, TX 76019

eMailSift: Adapting Graph Mining Techniques for Email Classification

Manu Aery and Sharma Chakravarthy

Technical Report CSE-2004-7
July 2004

eMailSift: Adapting Graph Mining Techniques for Email Classification

Manu Aery and Sharma Chakravarthy
IT Laboratory and CSE Department
The University of Texas at Arlington
{aery, sharma}@cse.uta.edu

July 29, 2004

Abstract

Text classification is the problem of assigning pre-defined class labels to incoming, unclassified documents. The class labels are defined based on a set of examples of pre-classified documents, used as a training corpus. For text classification, a number of approaches have been proposed such as Support Vector machines, Decision trees, k-nearest-neighbor classification, Linear Least Square fit and Bayesian classification among others. The need for handling and classifying large amounts of personal emails have prompted the use of text classification approaches to address email classification. Email classification is trickier than text classification as it is based on personal preferences, consequently it uses disparate criteria which are difficult to quantify. Also, documents are richer in content as compared to emails whose content can vary dramatically from folder to folder, hence conventional approaches may not be well-suited. In addition, as opposed to a static set of corpus typically used for training in text classification, the mail environment is constantly changing, with a need for adaptive and incremental re-training.

In this report we propose a novel, graph based mining approach for email classification. Our approach is based on the premise that representative – common and recurring – structures/patterns can be extracted from a pre-classified email folder and the same can be used effectively for classifying incoming email messages. A number of factors that influence representative structure extraction and classification are analyzed conceptually and validated experimentally. In our approach, the notion of inexact graph match is leveraged for deriving structures that provide coverage for characterizing folder contents. Extensive experimentation validate the selection of parameters and the effectiveness of our approach for email classification. We also compare the performance of our approach with the Naive Bayesian classifier.

1 Introduction

Information Technology has revolutionized the way data and knowledge is shared among users spread over different parts of the globe. Instant access to large amounts of information is available through the Internet. This ability to access large amounts of information also

entails a need for mechanisms that determine the relevance of information being accessed. One such mechanism for mechanism is text classification which allows one to retrieve information that falls into categories of interest. Text classification is the problem of assigning pre-defined class labels to incoming, unclassified documents. The class labels are defined based on a set of examples of pre-classified documents used as a training corpus. Text classification has traditionally been applied to documents in the context of information retrieval and there exists a large body of work on classification. For text classification, a number of approaches have been proposed and they include Support Vector machines (SVM) [15], Decision trees [2], k-nearest-neighbor (kNN) classification, Linear Least Square fit and Bayesian classification (NB). Of the many approaches, both SVM and kNN approaches have shown better performance in classifying text [23].

In the internet age, rapid dissemination of data and a quick method to apprise others of the information available, is possible by means of electronic mail. Electronic mail can be viewed as a special type of document as it is primarily text along with some identifying information unique to it (e.g., from, to, subject etc.). Email has evolved as a convenient means of communication between various parties. It is a fast, efficient and an inexpensive method of reaching out to a large number of people at the same time. This very reason is also the bane of electronic mail communication. The average user is often overwhelmed by the sheer amount of email sent and received. A large part of the Internet mail traffic comprises of unsolicited bulk email or spam as it is popularly, rather notoriously known. Users often find themselves expending large amounts of time and effort sifting through the mass of email messages and classifying them to their corresponding folders. A means for classifying email seems even more important considering the amount of time spent in processing email by individuals. The time spent in this task can be greatly reduced if traditional classification techniques can be adapted to email classification thereby automating the process with sufficient accuracy and efficiency. Certainly, the presence of an automated system can reduce the time to classify mails, search for a particular mail and retrieve relevant mails. An email classifier is one of the critical tools needed for the effective management of information in the Internet age.

Although email classification can be viewed as a special case of text classification, the characteristics of documents and emails differ significantly and as a consequence email classification poses certain challenges, not often encountered in text or document classification. Some of the differences and the concomitant challenges are:

1. Each users' mailbox is different and constantly evolving. Folder contents vary from time to time as new messages are added and old messages are deleted. A classification scheme that can adapt to varying folder characteristics is important.
2. Manual classification of emails is based on personal preferences and hence the criteria used may not be as simple as those used for text classification. This distinction needs to be taken into account by any technique proposed/used for email classification.
3. The information content of emails vary significantly, and other factors, such as the sender, group the email is addressed to, play an important role in classification. This is in contrast to documents which are richer in content resulting in easier identification of topics or context.

4. The characteristics of folders may vary from dense (more number of emails) to relatively sparse. A classification system needs to perform reasonably even in the absence of a large training set.
5. Emails even within a folder may not be cohesive. That is, the contents may be disparate and not have many common words or a theme. We characterize these folders on a spectrum of homogeneous to heterogeneous. A folder may lose its homogeneity as it becomes dense making it difficult to associate appropriate central theme/structures with the folder.
6. Emails are typically classified into subfolders within a folder. The differences in the emails classified to subfolders may be purely semantic (e.g., individual course offerings within the courses folder, travel within the projects folder etc.) or theme oriented. The ability to classify emails to appropriate subfolders will require clear separation of representative structures.

The above differences pose a number of issues for email classification. For example, the choice of representative structures need to consider homogeneity, folder size, average mail size, and other relevant characteristics specific to this domain. This entails extracting some of the characteristics by pre-processing the folders and using them effectively as parameters for the choice of training set, representative substructures, computing their rank etc.

The approach proposed herein is a novel one that takes the above differences into account and is based on the use of mining techniques for email classification. We believe that our approach can be tailored to both general text/document as well as email classification (using different graph representations as will be detailed later). It provides a general framework for using mining techniques and an approach to leveraging domain characteristics for classification.

1.1 Our Approach

Data Mining is the process of discovering interesting, non-trivial, implicit, previously unknown and potentially important information and patterns from data [13]. We propose a novel approach that adapts mining techniques for email and document classification. Supervised learning along with domain characteristics are exploited to identify the composition of previously labeled emails and these are used for the classification of incoming emails. Typically, users organize mail folders based on their content, certain patterns that occur in the mail messages, and personal preferences (for creating folders and sub-folders). Our approach is based on the premise that a folder consists of representative emails and the structure and content of these representative emails can be extracted by adapting graph based mining techniques to work with domain knowledge. We also hypothesize that the notion of inexact graph matching (or isomorphic graph comparison) is beneficial and helps in grouping structures within emails instead of looking for exact/identical matches that may be difficult in the email domain. In our approach, a folder is mined to obtain frequent and representative patterns using inexact graph matching guided by a threshold value. When a new mail arrives, it is matched with the set of patterns for each folder and in case of a match, a classification rank is associated for the incoming email with respect to the folder.

The actual classification is based on the rank (i.e., an email is classified into a folder for which it exhibits the highest rank).

The Subdue [11] substructure discovery algorithm is used to discover frequent and representative patterns corresponding to a given folder. Subdue discovers frequently occurring subgraphs using the minimum description principle [10]. Isomorphism or inexact graph match is used to make sure similar (and not merely exact) substructures are identified. Since Subdue identifies a large number of such patterns, they are sifted into a manageable number of patterns using several criteria, such as the frequency of occurrence, size of the pattern, average mail size, folder size and so on. The substructures are ranked and the set of representative substructures that best characterize the folder are chosen. Subsequently, any incoming email message is compared with the representative set of substructures of each folder. Again, inexact graph match techniques are used to determine the destination folder for the mail message. It is classified to the folder where the match is maximal. If no match occurs above a certain threshold, the mail is classified to a default folder and user-intervention may be needed to classify it.

One of the contributions of this report is an extensive evaluation of graph mining techniques and the effect of various parameters on classification. We analyze these parameters and provide experimental results to support the analysis. One of the goals is to establish a relationship among the factors relevant to a domain and their effect on graph mining in order to tune them effectively for classification.

The remainder of the report is organized as follows. Section 2 presents the related work in the areas of text and email classification. Section 3 gives an overview of graph based mining and a brief description of the Subdue substructure discovery system. An overview of the proposed eMailSift mail classification system is presented in Section 4. Section 5 explains the system architecture and its components along with the detailed discussion of the system and the parameters involved. Section 6 presents the experimental results and comparison with Naive Bayes approach. Section 7 has conclusions and future work.

2 Related Work

Text classification is the problem of assigning pre-defined class labels to incoming, unclassified documents. The class labels are defined based on a set of examples of pre-classified documents, used as a training corpus. For text classification, a number of approaches have been proposed and they include Support Vector machines (SVM) [15], Decision trees [2], k-nearest-neighbor (kNN) classification, Linear Least Square fit and Bayesian classification (NB). Of the many approaches, both SVM and kNN approaches have shown better performance in classifying text [23]. Schenker, Last et.al.,[21] have used graph models for classifying web documents. They use an extension of $k-NN$ algorithm to handle graph based data. The graph theoretical-distance measure for computing the distance translates to the maximal common subgraph distance proposed in [5].

Many text classification techniques have been applied to the problem of email classification. Based on the mechanism used, email classification schemes can be broadly categorized into: i) Rule based classification, ii) Information Retrieval based classification and iii) Machine Learning based classification techniques.

2.1 Work on Email Classification

Rule Based Classification Rule based classification systems use rules to classify mail messages into folders. William Cohen [9] uses the *RIPPER* learning algorithm to induce "keyword spotting rules" for email classification. *RIPPER* is a propositional learner capable of handling large datasets [8]. Cohen argues that keyword spotting is more useful as it induces an understandable description of the email filter. The *RIPPER* system is compared with a traditional IR method based on the *TF-IDF* weighting scheme and both show similar accuracy. *i-ems* [12] is a rule based classification system that learns rules based only on sender information and keywords. *Ishmail* [14] is another rule-based classifier integrated with the Emacs mail program Rmail. Although rules are easy for people to understand [6], managing a rule set may not be so. As the number and characteristics of incoming mails change, the rules in the rule set may have to be modified to reflect the same. This puts a cognitive burden on the user, to review and update the rule-set from time to time, often involving a complete re-writing of rules.

Most of the email managers (e.g., outlook, eudora), allow users' to set rules for classifying email to folders. These rules have to be specified manually and can use words from various categories. The main problem here is in the manual specification and management of these rules which can become cumbersome and need to be changed often to make them work properly.

Information Retrieval Based Classification Segal and Kephart [22] use the TF-IDF classifier as the means for classification in *SwiftFile*, which is implemented as an add-on to Lotus Notes. It predicts three likely destination folders for every incoming mail message. The TF-IDF classifier performs well even in the absence of large training data, and the classifier accuracy remains reasonable even as the amount of training data increases, adding to the heterogeneity of a folder. The classifier learns incrementally with every new message that is added or deleted from a folder, eliminating the need for re-training from scratch.

Machine Learning Based Classification Various machine learning based classification systems have been developed. The *iFile* system by Rennie [17] uses the Naive Bayes approach for effective training, providing good classification accuracy, and for performing iterative learning. The Naive Bayesian probabilistic classifier has also been used to filter junk mail effectively as shown by Sahami et.al[20]. The *Re:Agent* mail classifier by Boone [3] first uses the TF-IDF measure to extract useful features from the mails and then predicts the actions to be performed using the trained data and a set of keywords. It uses the nearest neighbor classifier and a neural network for prediction purposes and compares the results obtained with the standard IR, TF-IDF algorithm. Mail Agent Interface (*Magi*) by Payne and Edwards [18] uses the symbolic rule induction system *CN2* [7] to induce a user-profile from observations of user interactions. The system suggests actions such as 'delete', 'forward' and so on for each new mail message based on the training, hence results for multi-class categorization are difficult to assess.

The approach we propose in this report is different from the earlier approaches taken for text/document classification. It is also different from the few approaches that have been attempted for email classification. To the best of our knowledge, we are not aware of any

work on the use of graph mining techniques for email or text classification. One of the key contributions of this report is the adaptation of graph mining techniques for classification. Our preliminary results are very encouraging. Also, we believe that our proposed technique for email classification can be extended for any general purpose text/document classification.

3 Graph Based Data Mining

Graph based mining, as opposed to transactional mining (association rules, decision trees etc.), mines structural data. For applications that have an inherent structure (e.g., chemical compounds, proteins) graph mining is certainly appropriate as compared to other techniques as mapping them into other representations would lose the inherent structure. On the other hand, many applications, such as documents and emails can be represented using structural relationships thereby making them amenable to graph mining. As we will discuss later, the representation chosen is likely to be important and one of the challenges for applying graph mining is to map the data into an appropriate representation. Typically, entities, and relationships between them are used to form structures. Efficient mining techniques are required to discover structural similarities within these kinds of data. A graph representation is a natural choice for representing complex relationships between data objects. Entities are mapped to vertices and a relationship between them is represented by an edge between the corresponding pair of vertices. Graph based data mining aims at discovering these interesting and repetitive patterns/substructures within structural data. Relevant work in graph mining include the Subdue substructure discovery system by Cook and Holder [11], and the frequent subgraph approach by Kuramochi and Karypis [16] that maps the Apriori [1] algorithm to structural data represented in the form of a labeled graph and finds frequent itemsets that correspond to recurring subgraphs in the input.

3.1 Subdue Substructure Discovery System

The Subdue substructure discovery algorithm [11] is a graph based mining algorithm that discovers repetitive and interesting substructures in graph representations of data. It accepts as input a forest and identifies the best subgraph that minimizes the input graph using the minimum description length (MDL) principle. It outputs subgraphs of different sizes and their occurrence frequency in the input graph. Within the graph representation, entities and objects are mapped to vertices and the relationship between objects is represented as an edge between the corresponding pair of vertices. A substructure is a connected subgraph within the graph representation. An instance of a substructure in an input graph is a set of vertices and edges from the input graph that match, graph theoretically, to the graphical representation of the substructure(as defined in [11]). The problem of pattern discovery then reduces to the process of discovering subgraphs in the graph representation of data. Subdue is capable of identifying both exact and inexact (or isomorphic) substructures in the input graph. It uses a branch and bound algorithm that runs in polynomial time for inexact graph match and identifies graphs that differ by vertex or edge labels using a threshold parameter that characterizes the inexactness.

The aims of the discovery algorithm are two-fold, to discover repetitive and interesting substructures/patterns and to compress the original graph by abstracting instances of these

substructures. The substructure discovery process can be repeated over this abstracted data to provide a hierarchical or layered view of the original data in terms of the discovered substructures. The input to the Subdue substructure discovery system is a file with the list of unique vertices in the graph and the corresponding edges between them. The output generated by the system consists of a list of substructures that best compress the input graph.

3.2 Compression

Subdue uses two schemes for compression, the first based on the minimum description length (MDL) principle and the second based on size. The MDL principle has been used in various applications such as decision tree induction, image processing and others. The principle described by Rinssanen [19], states that the best theory to describe a set of data is one that minimizes the description length of the entire data set. Subdue uses MDL to determine the best substructure. The best substructure is one that minimizes the description length of the original input graph. Accordingly, the description length of the input graph is then $DL(S) + DL(G|S)$, where S is the discovered substructure, G is the input graph, $DL(S)$ is the number of bits required to encode the substructure, and $DL(G|S)$ is the number of bits required to encode the input graph G after it has been compressed using substructure S . If $DL(G)$ represents the number of bits required to encode the input graph G , the final MDL value of the graph is defined as

$$MDL = \frac{DL(G)}{DL(S) + DL(G|S)}$$

A higher MDL value signifies a substructure that reduces the description length of the original data or in other words, compresses it better. The compression value is defined as

$$Compression = \frac{1}{MDL}$$

The second compression scheme is based only on Size. This theory uses simple and more efficient method but it is less accurate as compared to the MDL metric. The value of a substructure S in graph G is

$$\frac{Size(G)}{(Size(S) + Size(G|S))}$$

Here, $Size(G) = \text{Number of vertices}(G) + \text{Number of edges}(G)$

$Size(S) = \text{Number of vertices}(S) + \text{Number of edges}(S)$

$Size(G|S) = (\text{Number of vertices}(G) - i * \text{Number of vertices}(S) + i) +$
 $(\text{Number of edges}(G) - i * \text{Number of edges}(S))$

where, G is the input graph, S is the substructure and $G|S$ is the input graph after it has been compressed by the substructure and i is the number of instances of the substructure.

3.3 Inexact graph Match

A substructure is a connected subgraph within the graph representation. An instance of a substructure in an input graph is a set of vertices and edges from the input graph that match, graph theoretically, to the graphical representation of the substructure. Though exact graph match comparison discovers interesting substructures, most of the substructures in the graph may be slight variations of another substructure. However, inexact graph match allows us to combine multiple structures into a single structure both for representation and identification. In order to detect substructures that match inexactly or vary slightly in their edge or vertex descriptions, the algorithm developed by Bunke and Allerman [4] is used where each distortion is assigned a cost. A distortion is defined as the addition, deletion or substitution of vertices or edges. The two graphs are said to be isomorphic as long as the cost difference falls within the user specified threshold. It is an exponential algorithm as it compares each vertex with every other vertex in the graph. Subdue uses a branch and bound approach, to consider a reduced number of mappings, that executes in polynomial time.

3.4 Subdue Parameters

Subdue accepts as input a number of parameters that not only affect substructure discovery, but also the set of substructures returned as best. The parameters that are relevant for our work are discussed below:

Threshold: Inexact graph match is one of the most important aspects of our approach. It allows for substructures that vary slightly in their vertex or edge label descriptions to be chosen as instances of a single substructure. The amount of variation permissible is determined by the *threshold* parameter. It specifies the bound on the difference that is allowed between instances of a substructure. Subdue assigns all transformations (insertion, deletion of an edge or vertex and so on) between instances a cost of 1. For a given substructure instance, to be classified as an instance of another substructure *sub*, the following condition needs to be satisfied: $matchcost(sub, inst) \leq size(inst) * threshold$. In other words, the total transformation cost needs to be less than the number determined by the particular value of threshold and substructure size.

If the size of the substructures is large, then even with a small value of *threshold*, there can be a large variation in the edge and vertex labels of the two instances being considered. The default value for *threshold* is 0.0, which means that the graphs have to match exactly. A very large value of *threshold* may not be meaningful as it will match two dissimilar graphs. Currently, Subdue supports *threshold* values up to 0.3. The exact value of *threshold* has to be determined from the size of the input graph, again knowledge of folder characteristics is essential for determining the same.

Nsubs: The input parameter *nsubs* specifies the maximum number of substructures that are returned by Subdue as the set of best substructures. For large graphs the number of interesting and repetitive substructures will be large as compared to graphs of smaller sizes. A low value of *nsubs* may not capture all the subgraphs that are necessary or meaningful for classification. So, this parameter needs to be derived using the folder (domain) knowledge.

Beam: Subdue employs a *beam* value during its search to determine the set of best substructures that it will carry forward from one iteration to the next. Substructure discovery is performed by expanding the vertices of the input graph by the edges that are incident on them. During each iteration of the algorithm, *beam* specifies the maximum number of substructures (the best substructures from all the iterations up to the current one) that are retained for expansion during the next iteration. With a low value of *beam*, potentially important substructures may be lost. A very high value of *beam* may retain substructures that are not very frequent and hence may not be of interest. The default value of *beam* used by Subdue is 4.

Minsize: The *minsize* parameter constrains all substructures that are picked as best substructures to have sizes equal to or larger than the specified parameter value. Size of the substructure here refers to the number of vertices in the substructure. In certain cases of input, smaller sized substructures may be very frequent, and for classification, smaller sized substructures that are prevalent across all classes, will serve no use for determining the distinctive substructures in each class. The substructure size can be constrained above a minimum size to pick substructures that contain something more than a common 'core'. The representation of the graph chosen may have a bearing on this parameter.

4 eMailSift: An Overview

As described in the previous section, Subdue is a graph based mining algorithm that accepts as input a forest, and identifies the best subgraph that minimizes the input graph using the MDL principle. In addition, it also outputs subgraphs of different sizes and their occurrence frequency in the input graph. As observed earlier, our premise is that we can adapt Subdue to extract/identify structures in the input folder and use them in various ways for classification. The other factor in our adaptation is the use of inexact graph match of Subdue and tune it to our need. Subdue is capable of identifying both exact and inexact (or isomorphic) substructures in the input graph. This is a critical aspect of our approach as we are not likely to find structures that are identical in an input folder. However, inexact graph match allows us to combine multiple structures into a single structure both for representation and identification. This will not only reduce the number of representative substructures that need to be maintained but also allow for comparison of multiple graphs in one match. We believe that this will be an important parameter that needs to be evaluated and tuned for our approach. The actual process of email classification in the eMailSift system consists of several steps including the representative substructure discovery process. A brief description of each step is given below. They are further elaborated in the following sections.

Folder/document pre-Processing: The email messages in a given folder form the training set used for generating interesting and repetitive patterns. Various characteristics of the mail folder need to be taken into account in order to derive the optimal set of parameters that are input to Subdue for substructure discovery. The average email size in a folder, the number of words retained from each email are examples of folder characteristics.

Graph Representation: The first step is to choose a graph representation that is appropriate for the email domain and use it for representing the email messages in a folder. Emails have a definite common structure in terms of their content (e.g., from, to, subject, body) and if need be the body can also be represented at a finer granularity. We will discuss two graph schemes, one that lends itself to any general text/document classification and the second that is tailored for email classification.

Substructure Extraction: The Subdue substructure discovery algorithm is used for extracting representative substructures. The substructures output by Subdue are determined by the parameters used including the threshold selected for inexact graph match.

Representative Substructure Pruning: The output generated by Subdue consists of a large number of substructures. Retaining all of them may not be necessary. Our goal is to identify the subset that is needed for discriminating incoming email during classification. In addition, the computation time is also affected by the number and size of graphs input to Subdue in addition to other parameters such as threshold. Hence, the output of Subdue is pruned based on a number of parameters to determine the representative set of substructures for the given folder. Those that are retained must be frequent in the folder and their sizes should be appropriate as well.

Representative Substructure Ranking: As a number of substructures (based on folder size, average email size etc.) will be retained after pruning, it is important to discriminate among them from the viewpoint of classification. Each representative substructure needs to be ranked in some way to indicate its representativeness. The rank associated is used in classifying incoming emails. Certainly, there is a difference between an incoming email matching a highly ranked substructure versus an average one. This will be important when incoming emails are classified into multiple folders.

Processing Input Email: Every incoming email message is converted into a chosen graph format for classification.

Classification: The representative substructures are compared with the incoming email message to determine if any of them appear in the incoming email graph. The processing step is explained in detail in the following section.

In this report, email classification will be considered for a single folder. We will use the cross-validation techniques to ascertain the success rate for a given folder. This will be later extended to classification of emails into multiple folders. For multiple folder classification, in the case of more than one match, the ranks associated with the matched substructures are used for resolving the folder to which it is classified. The email is classified into the folder with the highest ranked substructure match.

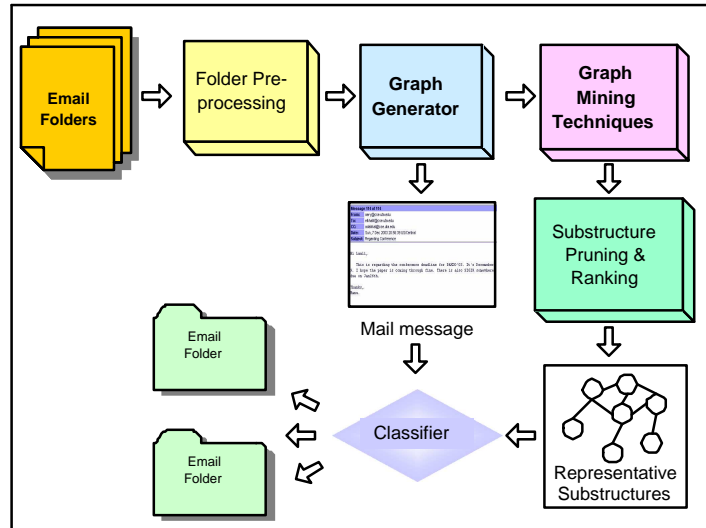


Figure 1: eMailSift

5 eMailSift: Architecture and Component Details

The eMailSift mail classification system uses graph based mining techniques to classify emails into the corresponding folders. The overall flow is shown in figure 1. A folder is pre-processed before generating a chosen graph representation for each email in that folder. The pre-processing step for most text mining tasks includes stop word removal and stemming. The current implementation includes only the elimination of stop words. Different criteria are used to choose the words for generating a graph from the emails in a folder. Subdue is then used on this graph to generate repetitive and interesting substructures. The resultant interesting and repetitive patterns generated by Subdue are further processed and pruned to identify the representative set of substructures for the folder. The representative substructures, once generated, are ranked for use in classification. This step is performed only once for each folder. As the folder evolves and its size and other characteristics change, new representative substructures need to be generated. This can be done either incrementally or by recomputing the substructures from scratch. We will be exploring incremental approach as part of this project. Finally, the incoming email message is converted into the same graph format used while generating graphs from folders. It is compared with the set of representative substructures of all folders and filed where the similarity is maximum. The classifier computes the similarity measure and resolves the destination folder in case of multiple matches. Each step in the classification process is elaborated below along with the rationale for the choice of parameters and alternatives considered.

5.1 Folder/Document Pre-processing

Using the entire folder or the document would be an overkill for any classification mechanism. Information retrieval uses several techniques for pre-processing documents to prune the size of the input without affecting the final outcome. For our purposes, we need to determine what can be pruned and what need to be retained.

Before generating the input graph for a folder, the emails in the folder are processed for stop word removal. Since the goal is to retain substructures that are frequent across emails in a given folder, the terms that comprise the substructures have to be frequent as well. These terms must occur frequently, preferably across all emails in a folder and not merely in a single email. This choice of retaining words that are common across mails takes care of the disparity in mail length, as some emails are considerably longer than others. Words are ranked based on their occurrence frequencies across all mails in a folder and those whose frequencies account for more than $f\%$ of the sum of all frequencies are retained. The rationale behind this is to study the effect of f on classification. Of course, by choosing f to be 100, one would retain all the words. But, the lower frequency words may not contribute towards classification. The parameter f need be tuned to observe its effect, and identify any possible dependency on other folder characteristics. Currently, this parameter is set to 80%. To construct the graph, only those words in the message body that are a member of this frequent set are used. This ensures the words chosen are frequent not only in a single email, but across a substantial number of emails in the folder under consideration.

The mail messages in a folder form the training set for the classification process. All the mails in the folder are pre-processed as explained above and converted into a graph (actually a forest) for substructure discovery. The graph that serves as input to the Subdue algorithm is a sequence of vertex and edge descriptions. For our experiments, we have used a 80-20 and 60-40 split for cross validation. That is either 80 or 60 percent of the emails are randomly chosen from a folder as the training set and 20 or 40 percent are used for classification.

5.2 Graph Representation

Graph Representation Schemes The first step in discovering representative patterns is to transform the email messages into a graph format for mining. The choice of a graph representation can have an impact on the classification as structures identified will have different set of relationships. Also, based on the domain knowledge, it is possible to choose a representation that provides different emphasis for different domains. For example, if we want to give more emphasis to the header of an email, then we can choose a representation to reflect that. We have explored alternate graph representations for this domain.

As we believe that our current approach can be useful for general document classification, one of the schemes proposed is a generalized graph representation that can be used for any kind of text document. The second scheme is specifically aimed at representing email messages in graph format. One way to classify a document is in terms of the important words in that document. A simple graph representation is to use a star-graph. It consists of a central anchor or root vertex and the chosen words from the document form the remaining vertices, along with edges that connect them to the central root vertex with a descriptive edge label such as *contains*. The ability to label edges makes this simple representation

quite effective even for emails if the labels correspond to the various components of an email. Figure 2b shows such a representation where the edge labels used are *from*, *subject*, *to*, and *contains* to differentiate between various significant components of an email.

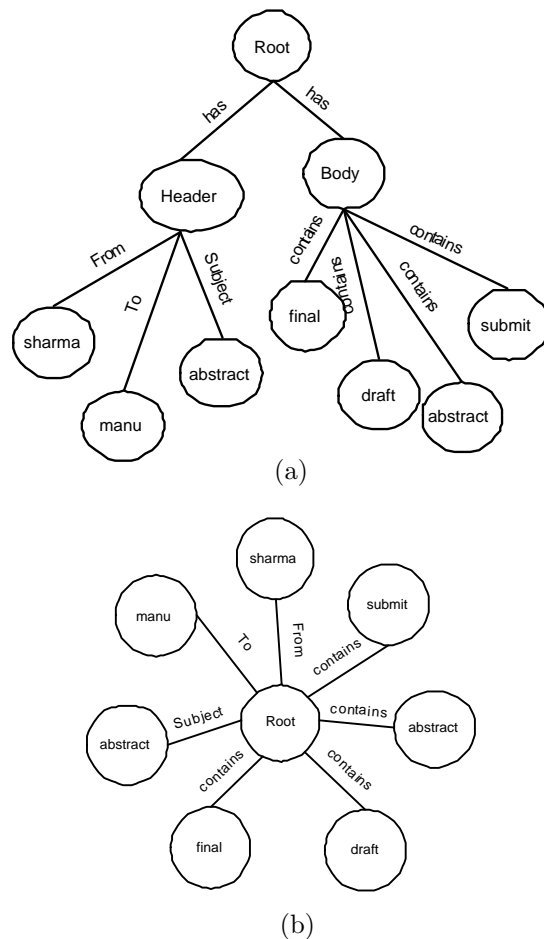


Figure 2: Graph Representations

Of course other representations that use the components of an email explicitly can be formulated. Scheme (a) in the figure 2 corresponds to the scheme specifically customized for representing emails as a graph. The representation differentiates between the email headers and the contents of the email body. It was devised to study classification using either headers only or all the information present in the email message. Matches in the email headers can be given a different weight and considered superior to the matches in the message body. As shown, the scheme (b) in the figure can be used either for for general text/document classification or email classification. For email classification, this scheme considers all the

information in a message, with each word in the email connected to the central root vertex in a star-like configuration. All the experiments results reported are with this representation scheme.

The input file to the Subdue system consists of vertex and edge entries. Each vertex entry associates a unique vertex id with every vertex label. Each entry corresponding to an edge represents an undirected edge between a pair of vertices and the associated edge label. The input file to the Subdue system corresponding to the representation in figure 2b is shown in figure 3.

```
v 1 Root
v 2 sharma@cse.uta.edu
v 3 manu@cse.uta.edu
v 4 abstract
v 5 final
v 6 draft
v 7 abstract
v 8 submit
u 1 2 From
u 1 3 To
u 1 4 Subject
u 1 5 contains
u 1 6 contains
u 1 7 contains
u 1 8 contains
```

Figure 3: Input Graph File

5.3 Impact of Folder Characteristics

Our goal is to identify representative structures of a given folder. In order to achieve this, we have to choose a number of input parameters for the Subdue algorithm (such as *beam*, *threshold*, etc.). We believe that the folder itself needs to be used as a source for deriving these parameters as folders vary in their characteristics. This way, the representative structures generated will be customized to a folder and not based on a fixed set of parameter values. To determine the representative substructures that best characterize a particular folder, certain characteristics of the folder need to be derived. We need to identify the parameters that are important and then provide a way for deriving them with substantiation. These parameters must be tunable and be effective for diverse email and folder characteristics. Not all folders in a email hierarchy exhibit similar properties. Certain folders may be more dense as compared to others, providing a greater amount of information for training the classifier. Emails in a particular folder may exhibit greater cohesion than those in other folders, making it easier to determine the similarities among them. Due to constant addition, deletion and movement of emails across folders, the folder contents change with time. These and other folder characteristics need to be quantified and specified as input parameters to the Subdue discovery algorithm to ensure that the substructure discovery process is

based on the traits of the folder. If the discovery process is guided by these parameters, the representative substructures generated is likely to better reflect the contents of the folder. Below, we discuss the key characteristics of email folders that we believe affect substructure discovery and how they can be mapped as parameters for the same.

5.3.1 Average email Size and Threshold

As discussed earlier, the *threshold* parameter determines the amount of inexactness allowed during substructure discovery. During inexact graph match, threshold determines the number of vertices and edges that vary among instances of the same substructure. The actual number is determined by

$$(num\ of\ vertices + num\ of\ edges) \times threshold \quad (1)$$

Since each email in a folder has different size (even after pre-processing), we use the average email size of a folder as one of the characteristics of a folder. We argue that a low value of *threshold* allows for a significant amount of inexactness while comparing substructure instances for email messages that contain a greater a number of words in the message body. This is because for a large email body, even with a low *threshold*, the actual value as determined by equation 1 will allow a reasonable amount of inexactness. This ensures that similar substructures with slight variations are identified. For emails with smaller message bodies and hence fewer vertices in the input graph representation, a larger value of threshold is required for inexact match. Using the size of the email messages in a folder, we can always determine the amount of inexactness to allow for graph match. If the amount of inexactness to be allowed in terms of the number of edge/vertex label variations is '*i*', then the particular value of threshold is obtained by

$$threshold = \frac{i}{Avg_{ms}} \quad (2)$$

Here, we have interpreted the average email size as a parameter that affects pattern discovery and used it to derive the right value of the threshold that allows a reasonable amount of variation and at the same time preserves the similarity between substructure instances. It is also important to make sure that the value of *i* is not very large allowing very dissimilar substructure instances to be grouped as identical. A value of *i* more than 10 may not be appropriate even for a folder with a large value of average email size.

5.3.2 Average email Size & Folder Size Vs Number of Substructures

The number of structures returned by Subdue as the set of best substructures is limited by the parameter *nsubs*. To ensure that the representative set consists of substructures that characterize the folder the number of substructures to be returned by the discovery process has to be derived from the folder characteristics. If the average size of the emails in a folder is large, then the number of subgraphs is also large. Similarly, if there are a large number emails in a folder, then again, there are potentially a larger number of substructures that characterize the folder. Both of these folder characteristics seem to be important and need to be taken into account to determine the number of substructures generated for a given folder. Careful analysis reveals that number of substructures depends more upon the

average mail size than the number of email messages in the folder. This is due to the fact that an increase in the folder size may serve to increase substructure instances rather than the number of distinct substructures themselves. Whereas increase in the average size of an email folder increases the number of substructures significantly. We have derived the number of substructures by using both the folder size and the average email size along with weights that can be adjusted to less or more emphasize each factor as given by the following equation

$$nsubs = w_1 \times F_s + w_2 \times Avg_{ms}, w_1 < w_2 \quad (3)$$

where, F_s is the size of the folder and
 w_1 is the weighting factor applied to the same
 Avg_{ms} is the average size of the messages in the folder
 w_2 is the weight applied to the average email size.

The particular numeric values of w_1 and w_2 have been chosen as 0.5 and 0.75, and as can be ascertained, the greater influence of average email size is characterized by the higher weight value.

5.3.3 Beam Size

As *beam* determines the number of best substructures retained at the end of each iteration, a low value of *beam* results in losing out on some of the representative substructures. Due to small *beam* size some interesting substructures do not get a chance to be evaluated in subsequent iterations of the discovery algorithm. Hence, potentially interesting substructures may be lost. Similarly, a high value of *beam* retains substructures that are instances of the best substructure for the current iteration of Subdue. Also, substructures that are not too interesting are retained. Experiments employing *beam* sizes of 2, 4, 6, 8, and 10 were performed on the experimental folders. In almost all cases *beam* values of 6,8, and 10 picked the same set of substructures, with most of them being variations of the best substructure. With a *beam* width of 2 some substructures were not reported. *Beam* size 4 seems to be adequate return all the important substructures for folders that we have been experimenting with. The same has been retained for experimental purposes.

5.3.4 Substructure Size Vs. Minsize

From the graph representations of figure 2, it can be inferred that for scheme (a) the smallest sized substructure will contain at least four vertices. These are the root, header, and at least two among the 'To', 'From', 'Sender', 'Cc' address fields and the 'Subject' field if all the other fields are different. Substructures smaller than this are common to all emails within a folder and also across all folders. Therefore, we constrain the minimum size of the substructures reported to be four. In fact, using a lower *minsize* may affect the classification adversely. This constraint needs to be determined from the graph representation scheme employed. Applying the same argument, for representation (b) of the same figure, the minimum size of the substructure is three. The constraint is specified using the *minsize* parameter that is input to the Subdue system. This ensures that substructures that are picked have sizes greater than the size of substructures that are most likely to be common across a lot of email folders, and hence capable of discriminating between the same.

5.3.5 Substructure Pruning and Ranking

Subdue identifies a large number of structures along with their frequencies. One alternative is to retain all the structures produced by Subdue. However, retaining several structures with slight variations and having the same frequency will not contribute to classification especially when inexact match is used. To identify the representative set of substructures of a folder, pruning is carried out on the resultant substructures reported by Subdue. This is carried out by applying some metrics to determine the set of representative substructures that best characterize the folder under consideration. Due to inexact graph match, some substructures are similar to others in terms of occurrence frequency and substructure size, while varying in one or more vertex or edge descriptions. To prune this possibly large set of substructures, only those substructures that differ in the aforementioned characteristics are retained. This ensures uniqueness of the substructures. For instance, two substructures with six vertices each and different occurrence frequency are obviously different, as the same substructure will not be reported twice with different frequencies. Therefore each substructure in the representative set refers to a unique pattern that follows from the emails of the folder under consideration.

Since Subdue uses compression as a heuristic it also identifies certain large substructures that do not occur very frequently. These substructures do not significantly add to the representative set primarily because they do not cover a substantial portion of the email folder. Substructures with very low frequencies as compared to the folder size are therefore eliminated from consideration as potential representative substructures. The representative set of substructures generated after pruning are then ranked according to the equation given below and used for classification.

The substructures are ranked based on the frequency of occurrence, average email size, number of email messages in the folder and substructure size. Substructures with sizes comparable to the average email size and which cover a reasonable number of email messages are preferred, as this signifies correlation with folder contents. The relative frequency of occurrence of a substructure 's' is the ratio of f_s , the occurrence frequency of the substructure and N , the number of email messages in the folder. The size of the substructure, S_s , is normalized with the average size Avg_{ms} , of the emails in the folder. The rank R_s of each substructure is computed by the following equation

$$R_s = \frac{S_s}{Avg_{ms}} \cdot \frac{f_s}{N} \quad (4)$$

The rank computed above reflects the need for discovering substructures that not only cover a significant percentage of the input set, but also compare well with the average size of the emails. Relatively large sized frequent substructures signify greater similarity among emails of a folder. The ranking criteria becomes effective when an email matches more than a single folder. An incoming email that matches a substructure with a high value of rank in folder F_1 as compared to an average ranked substructure in folder F_2 , is rightly classified to F_1 .

5.4 Classification

The best n substructures that are chosen as representative substructures are ranked as explained previously. Obviously, n varies with the folder and average email size. These

ranked representative substructures are then used for classification. Each incoming email is compared with the set of ranked representative substructures of all folders. As with the generation of representative pattern subgraphs, inexact graph match is used for comparing the incoming email with the predefined representative substructures. The email is filed to the corresponding folder if a match occurs. Since, the representative substructures of each folder are ranked, in case of multiple matches, the email is filed to the folder with the highest ranked substructure signifying higher correlation with the folder contents.

6 Experimental Evaluation

6.1 Implementation

The eMailSift system has been implemented in Perl. The input to the system consists of one or more folders and a number of parameters such as the split for cross validation, choice of graph representation, beam value etc. The prototype pre-processes the folders, generates graphs, computed the various parameters using the folder and invokes the Subdue discovery algorithm. The output of Subdue is pruned and ranked for use in classification. The prototype also generates the graphs for the training set and executes the classification step. The outcome along with a number of other values generated are logged for further analysis.

The experiments have been carried out on Pentium Xeon 2.66 Ghz dual processor machines with 2GB memory. Extensive experiments on a large number of folders with diverse characteristics (i.e., different average mail size, dense, sparse folders, homogeneity and so on) have been carried out to study the effect of the various parameters on classification. Cross-validation was used for training and testing purposes.

6.2 eMailSift Vs Naive Bayes

Extensive experiments were carried out to compare the performance of eMailSift with that of the probability based Naive Bayesian classifier. Naive Bayes classifier predicts the correct class for an unknown sample based on its feature vector. Despite the simplifying assumption of term independence that is made, the classifier does fairly well. The performance of the classifier with exact graph match is compared with the Bayesian classifier in Figure 4. The top plot shows a 80/20 split of training/test set. The bottom plot shows the same for a 60/40 split.

We have used seven folders whose total size varies from 22 to 375 to ascertain that the classification is not specific to a particular folder or a particular size of the folder. In addition, these folders were selected from public listserv's and personal emails to provide diversity in the way they have been classified by humans. For the 80/20 plot (upper graph), it is interesting to note that naive fails completely for some folders (F1 and F4) where as eMailSift fails only for one folder (F1). This folder happens to be a small one (size 22) and the average email size is also small. The exact match in conjunction with small folder size might be the reason for this result. In cases where the naive Bayes performs better, it is only marginally better, whereas eMailSift's performance is comparable throughout and very good for certain folders.

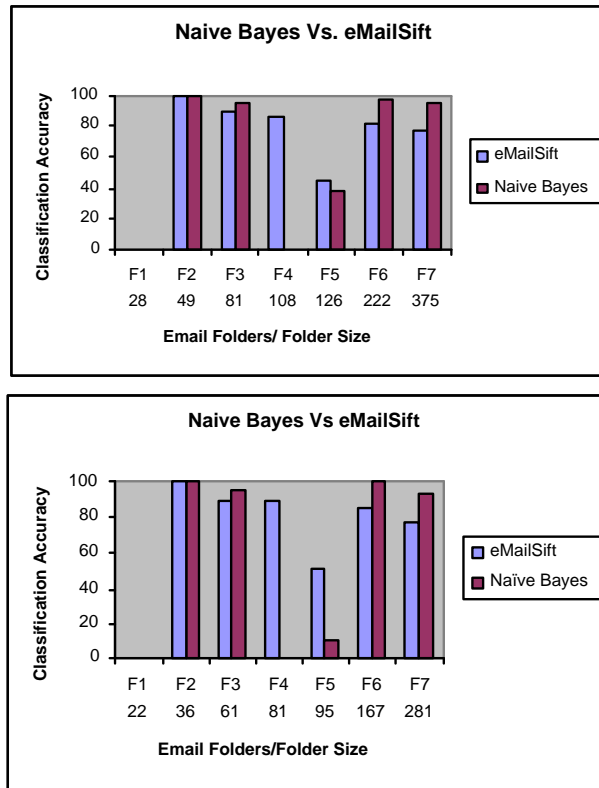


Figure 4: Performance of eMailSift Vs Naive Bayes

For the 60/40 split as training/data set, the accuracy of eMailSift has improved for some folders. Even in the case of reduced training data, as evident from the lower plot of figure 4, eMailSift performs much better than Naive Bayes. eMailSift was consistent in classifying incoming emails, but Naive Bayes was unable to classify even a single email for a couple of input folders. Our conjecture for improved performance with lesser training data is that the additional emails diluted the training set and affected discovery of substructures that characterized the folder contents.

When inexact graph match is used for training and classification, eMailSift far outperforms the probability based classifier. An interesting observation was eMailSift never classified an email as belonging to the 'wrong' folder, as opposed to Naive Bayes, which had a high rate of false positives. In our case, this was totally eliminated which is a significant achievement.

6.3 Exact Graph Match Vs Inexact Graph Match

Exact graph match works well with fairly good classification accuracy. Inexact graph match improves upon the classification as is evident from the graph of figure 5. For sparse folders, usage of inexact graph match enabled successful classification of test emails as opposed to Naive Bayes and exact match, which did not do well on sparse folders. This reinforces our argument for using inexact graph match to better classification accuracy.

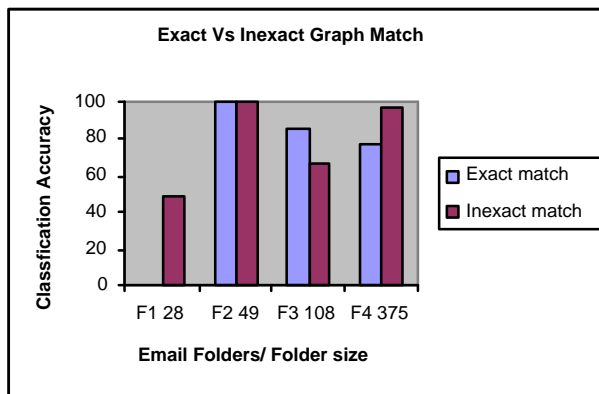


Figure 5: Inexact Graph Match Vs Exact Graph Match

6.4 Classification Accuracy Vs Folder Size

The eMailSift classifier works well on folders of all sizes. Even with an increase in folder size, leading to an increase in the heterogeneity of a folder, the classification accuracy remains good as shown in the graph of figure 6. An increase in training size is reflected as a variation in folder characteristics: folder size and average email size. The computation of these parameters and their subsequent interpretation in determining the parameters for substructure discovery ensures that the representative structures returned cover a significant amount of the similarities present in a folder. Heterogeneity of the folder contents with accumulation of mail therefore does not pose a problem. The same argument is true with a shrinking of folder contents. In essence, since the representative similarities of a folder are derived from the folder characteristics themselves, any change in the folder contents will lead to a corresponding change in the representative set. Hence, we ensure during classification the representative set used characterizes the current folder contents.

6.5 Pruning Vs Classification Accuracy

Experiments were carried out to observe the effect of pruning for determining the representative set of substructures. Classification of incoming emails was performed using a pruned set of representatives and also by retaining all substructures returned by the substructure

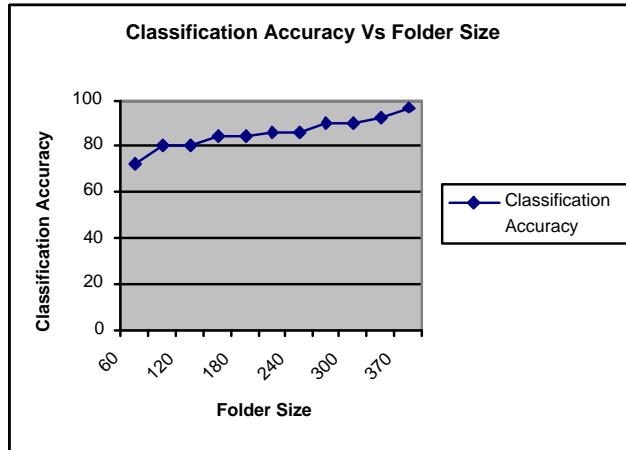


Figure 6: Folder Size Vs Classification Accuracy

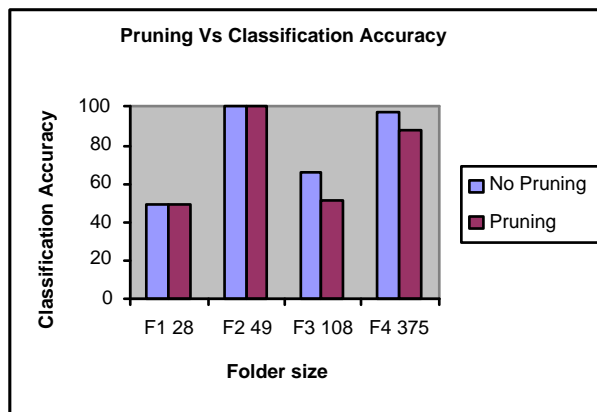


Figure 7: Pruning Vs Classification Accuracy

discovery process. As is evident from figure 7, pruning of substructures has no adverse affect on classification. This is due to the fact that the pruned substructures are those that do not occur very frequently and those which are covered by similar inexactly matching substructures with slight variations. Pruning helps in reducing the large set of representative substructures without any noticeable difference in classification accuracy.

7 Conclusions and Future Work

In summary, we have proposed an innovative approach for electronic mail classification using graph based mining. We believe that email folders consist of emails with similar patterns, and these patterns can be derived and used for classifying incoming emails. Data mining techniques that look for patterns in underlying data can be used to automate the process of pattern discovery and we have employed a graph based technique for the same. We have presented our analysis of the parameters and preliminary results that look promising. We believe the technique is applicable for general document and we page classification as well and intend to extend our work in that direction. We are currently evaluating alternative graph representations and systematically evaluating the effect of various parameters to draw general conclusions about the parameters applicable to a domain.

References

- [1] R. Agarwal and et.al. Mining association rules between sets of items in large databases. *Proceedings of 1993 ACM SIGMOD Conference*, 1993.
- [2] C. Apte, F. Damerau, and S. M. Weiss. Text mining with decision trees and decision rules. *Conference on Automated Learning and Discovery*, 1998.
- [3] G. Boone. Concept features in re:agent, an intelligent email agent. *Proc. Agents*, pages 141–148, 1998.
- [4] H. Bunke and G. Allerman. Inexact graph match for structural pattern recognition. *Pattern Recognition Letters*, pages 245–253, 1983.
- [5] H. Bunke and K. Shearer. A graph distance metric based on maximal common subgraph. *Pattern Recognition Letters*, pages 753–758, 2001.
- [6] J. Catlett. Megainduction: A test flight. *International Conference on Machine Learning*, 1991.
- [7] P. Clark and T. Niblett. The cn2 induction algorithm. *Machine Learning*, pages 261–283, 1989.
- [8] W. W. Cohen. Fast effective rule induction. *Proceedings of the twelfth International Conference*, 1995.
- [9] W. W. Cohen. Learning rules that classify e-mail. *Proceedings of AAAI-1996 Spring Symposium on Machine Learning in Information Access*, pages 124–143, 1996.

- [10] D. J. Cook and L. B. Holder. Substructure discovery using minimum description length and background knowledge. *Journal of Artificial Intelligence Research*, pages 231–255, 1994.
- [11] D. J. Cook and L. B. Holder. Graph based data mining. *IEEE Intelligent Systems*, 15(2):32–41, 2000.
- [12] E. Crawford, J. Kay, and E. McCreath. Automatic induction of rules for e-mail classification. *Proceedings of the Sixth Australasian Document Computing Symposium, Coff's Harbour, Australia*, 2001.
- [13] W. Frawley, G. Piatetsky-Shapiro, and C. Matheus. Knowledge discovery in databases: An overview. *AI Magazine*, 1992.
- [14] J. Heffman and C. Isbell. Ishmail: Immediate identification of important information, at&t labs. 1995.
- [15] T. Joachims. Text categorization with support vector machines: Learning with many relevant features. *ECML*, pages 137–142, 1998.
- [16] M. Kuramochi and G. Karypis. Frequent subgraph discovery. *IEEE International Conference on Data Mining*, pages 313–320, 2001.
- [17] J. D. M.Rennie. ifile:an application of machine learning to e-mail filtering. *Proceedings of KDD-2000 Text Mining Workshop, Boston Aug*, 2000.
- [18] T. Payne and P. Edwards. Interface agents that learn: An investigation of learning issues in a mail agent interface. *Applied Artificial Intelligence*, pages 1–32, 1997.
- [19] J. Rissanen. Stochastic complexity in statistical enquiry. *World Publishing Company*, 1989.
- [20] M. Sahami, D. Heckerman, and E. Horovitz. A bayesian approach to filtering junk e-mail. *AAAI-98 Workshop on Learning for Text Categorization*, 1998.
- [21] A. Schenker, M. Last, H. Bunke, and A. Kandel. Classification of web documents using a graph model. *Seventh International Conference on Document Analysis and Recognition*, 2003.
- [22] R. B. Segal and J. O. Kephart. Swiftfile: An intelligent assistant for organizing e-mail. *Proceedings of AAAI 2000 Spring Symposium on Adaptive User Interfaces*, pages 107–112, 2000.
- [23] Y. Yang and X. Liu. A re-examination of text categorization methods. *ACM SIGIR*, pages 42–49, 1999.