2010 IEEE International Conference on Data Mining

A Graph-Based Approach for Multi-Folder Email Classification

Sharma Chakravarthy
Department Of Comp. Sci. & Engg.
Univ. of Texas at Arlington
Arlington, TX, USA
sharmac@uta.edu

Aravind Venkatachalam
Department Of Comp. Sci. & Engg.
Univ. of Texas at Arlington
Arlington, TX, USA
avenkata@uta.edu

Aditya Telang
Department Of Comp. Sci. & Engg.
Univ. of Texas at Arlington
Arlington, TX, USA
aditya.telang@uta.edu

***Abstract*—This paper presents a novel framework for multi-folder email classification using graph mining as the underlying technique. Although several techniques exist (e.g., SVM, TF-IDF, n-gram) for addressing this problem in a delimited context, they heavily rely on extracting high-frequency keywords, thus ignoring the inherent structural aspects of an email (or document in general) which can play a critical role in classification. Some of the models (e.g., n-gram) consider only the words without taking into consideration where in the structure these words appear together. This paper presents a supervised learning model that leverages graph mining techniques for multi-folder email classification. A ranking formula is presented for ordering the representative - common and recurring - substructures generated from pre-classified emails. These ranked representative substructures are then used for categorizing incoming emails. This approach is based on a global ranking model that incorporates several relevant parameters for email classification and overcomes numerous problems faced by extant approaches used for multi-folder classification. A number of parameters which influence the generation of representative substructures are analyzed, re-examined, and adapted to multiple folders. The effect of graph representations has been analyzed. The effectiveness of the proposed approach has been validated experimentally.**

I. Introduction

Electronic mail is a fast, efficient, inexpensive and one of the most preferred way of communication among a large group of people. Emails can be viewed as a special type of document with some unique identifying characteristics such as *From* header, *To* header, *CC* header, *Attachments*, *Subject* etc. Although, email has become ubiquitous, management and maintenance of emails has become a major hassle. Most users are overwhelmed with a large amount of emails received or sent, and hence end up spending a large amount of time in sifting and classifying them to corresponding folders. A misclassification of an email is as good as losing the email considering the sheer volume of emails received each day and the number of folders to be maintained.

Automated classification of email be a savior. However, the characteristics of emails differ significantly from that of a document 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: i) classification of emails is based on personal preferences, ii) each mailbox is different and is constantly evolving; folder contents vary from time to time , iii) 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, iv) the characteristics of folders may vary from dense (more number of emails) to relatively sparse, v) emails within a folder may not be cohesive i.e., the contents may be disparate and not have many common words or a theme, and vi) emails are typically classified into subfolders within a folder.

To sum up, manual email classification uses disparate criteria which are extremely difficult to quantify. Hence conventional approaches may not be appropriate for email classification. The above characteristics also indicate the need for a supervised approach to email classification and the need for extracting commonly occurring and representative emails from folder that can be used for the classification of incoming email. In addition, as opposed to a static set of corpus typically used for training in text classification, the email environment is constantly changing with a need for adaptive and incremental re-training.

Information retrieval mechanisms [1], Machine learning [2], and Rule-based classifiers [3] are some of the common techniques applied to the problem of email classification. Most of these techniques rely only on the content and do not use any embedded structure for classification. They extract keywords or highly frequent words and ignore the importance of extracting a group of related words that co-occur in a context. Moreover, they fail to take into account the differences between an email and a normal text document and hence not utilize the characteristics of email for classification. They also fail to take advantage of the structural characteristics provided by an email.

The content in an email (or any document) has an inherent structure and reducing it to non-structural (or transactional) format [4] results in loss of information. On the other hand, a graph representation provides a more natural format for preserving the inherent structural characteristics. If process-

1550-4786/10 $26.00 © 2010 IEEE
DOI 10.1109/ICDM.2010.55

ing can be done on this representation, it is likely to provide better results as the semantics of the applications (in the form of relationships) is preserved during processing. Complex structural relationships can be modeled as graphs if no constraints are assumed (such as no cycles, no multiple edges, only directional edges, and constraints on vertex and edge labels). Unlike transaction mining, a graph representation comes across as a natural choice for representing complex relationships as the data visualization process is relatively simple as compared to a transactional representation. Data representation in the form of a graph preserves the structural information of the data which may otherwise be lost if it is translated into other representation schemes.

Our approach to utilize the structure of a document is unique from other forms of document classification that assume the document as a set (or bag) of words having no particular structure. Hence, we believe that designing a classification system that represents an email as a *graph*, instead of just a bag of words, has a better scope for achieving higher accuracy as compared to some of the conventional approaches. Hence, moving away from existing classification techniques, we adopt graph-based data mining for addressing the problem of email classification.

The novelty of the proposed approach is in leveraging graph mining techniques for multi-folder email classification. To the best of our knowledge, this is the first attempt to assess the applicability of graph mining for classification across multiple folders. The global ranking formula for ordering the representative substructures is another important contribution of this paper. Additionally, this paper performs an extensive evaluation of graph mining techniques and their effect on various parameters used for classification and determining the values of these parameters both analytically and experimentally. Extensive experimental evaluation has been performed to establish the soundness of the approach.
Road map: In Section II, we briefly survey the related work. Section III explains the proposed *m-InfoSift* framework and elaborates the details on pre-processing and graph representations. Section IV explains the processes of extracting and pruning representative substructures from these graph representations. In Section V, we explain our global ranking for substructure ordering, and elucidate the classification process. Section VI elaborates the experimental analysis of our approach, and Section VII concludes the paper.

II. Related Work

The range of techniques currently adapted for addressing the broader problem of document classification vary from Support Vector Machines (SVM) [5], Decision Trees [6], k-Nearest-Neighbor (k-NN) classifiers [7], Linear Least Square fit technique [8], Rule Induction [9], Neural Networks [10] and Bayesian Probabilistic Classification [4], [11].

Specifically, usage of TF-IDF for email classification has been proposed for the *SwiftFile* [1] system, as an add-on to Lotus Notes. To classify an incoming email, the term frequency vector of the email is constructed. It is compared with the TF-IDF vectors of all folders using a cosine similarity metric, and classified to the folder where the value of the cosine distance function is maximum.

The *iFile* system [2] 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 effectively filter junk email in [12]. The *Re:Agent* email classifier [13] uses a combination of the TF-IDF measure to extract useful features from the mails, and the nearest neighbor classifier and a neural network for prediction purposes and compares the results obtained with the standard TF-IDF algorithm. On similar grounds, the *MailCat* [14] system proposes the use of text classifiers for organizing emails.

Rule-based classification systems use rules to classify emails into folders. Mail Agent Interface (*Magi*) [15] uses the symbolic rule induction system *CN2* [16] to induce a user-profile from observations of user interactions. [3] uses the *RIPPER* learning algorithm to induce *keyword spotting rules* for email classification. The *i-ems* (Intelligent Mail Sorter) [17] rule based classification system learns rules based only on sender information and keywords. *Ishmail* [18] is another rule-based classifier integrated with the Emacs mail program Rmail.

Kiritchenko et.al., [19] employs temporal features (e.g., day of the week, time of the day, etc.) in order to classify email messages into classes. Relevant temporal features are extracted from emails and combined with conventional content-based classification approaches to build a much richer information space to improve accuracy. The *EMail-Valet* [20] proposed text classifiers by adding numerical to obtain a flat graph representation for classifying emails. Graph mining techniques for binary classification of documents in a delimited context has been proposed by Aery et.al., [21]. However, in this paper, we propose a generalized approach for establishing a comprehensive multi-folder classification framework using graph mining.

III. The m-InfoSift Framework

Figure 1 shows the architecture of the *m-InfoSift* framework. Similar to any supervised learning process, it uses two phases to classify emails: *Training* and *Classification*. The central idea is that a pre-classified folder (or parts of it) can be used to extract substructures that represent a folder accurately. They will be ranked globally and used for classifying unknown emails. Since emails are not likely to match exactly, the use of inexact match with tight bounds is used to check for closeness. We use Subdue [22] for substructure extraction and inexact subgraph matching.

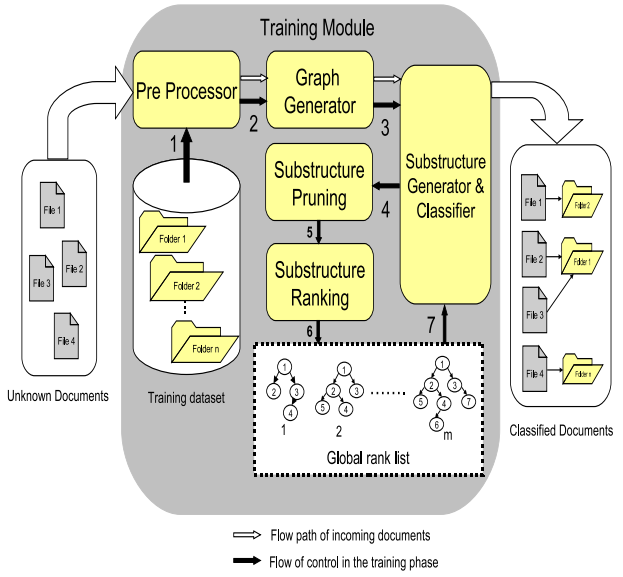

Figure 1. m-InfoSift System Overview

A. Pre-processing

A document usually contains a number of unnecessary words that can adversely affect the classification of the document. Several techniques have been used for document pre-processing to prune the size of input to retain only interesting words. The main goal of pre-processing in *m-InfoSift* is to facilitate the retention of frequent substructures across the document. Thus, all the words that comprise the substructures have to be retained in the document as well. The words have to occur frequently across all documents instead of a single one. This notion of retaining the frequent words across the documents takes care of the disparity of some documents being longer than others. Therefore, prior to representing the documents as graphs, the documents are pre-processed as follows:

Stop-word Elimination: Words such as conjunctions, articles and even common words that occur frequently across all documents are eliminated. These are generally regarded as 'functional words' which do not carry meaning. The assumption is that their meanings can be interpreted by ignoring these functional words.

Stemming: It is the process of reducing the inflected words to their roots/base/stem. This process reduces the number of unique words through out the documents and also aids in classification. For example, the words '*seeing*', '*see*', '*seen*' are all reduced to the same parent word – '*see*'. Similarly, words ending with 'ed', 'ing', 'ly', which are used to represent the tenses of the verb or adjectives in English grammar, are stripped to their root. A problem with Stemming might be *homograph disambiguation* i.e., single word with multiple meanings. For example, the word 'saw', which would be reduced to its root 'see', as in the previous example, can also mean a wood-cutting tool. Since the advantages of stemming surmounts its limitations, it is followed as a part of our preprocessing.

Feature Selection: It [23] is a technique commonly used in machine learning for selecting a subset of relevant features in order to build the learning model. This process removes the most irrelevant and redundant features from data and also helps improve the performance of learning models by enhancing the generalization capability and ameliorates the learning process. In our system, words, after Stop-word elimination and Stemming, are ranked based on their occurrence frequencies across the documents in a class and only those words whose frequencies account for more than *f%* of the sum of all frequencies are retained. Only those words that are a part of this frequent set are considered for generation of graphs. The intuition behind this being that lower frequency words are not likely to contribute towards classification. The parameter f has been tuned experimentally by observing its effect on classification. This ensures the words chosen are frequent not only in a single document, but across a substantial number of documents in a class.

B. Graph Representation

Once the test documents have been pre-processed and the respective folder characteristics computed, the documents are represented in the form of graphs using the **Graph Generator** module. The inherent structure of emails is used to generate graphs. The graph representations are chosen based on the domain knowledge in order to provide emphasis on the domains (e.g., structure of an email message). We have proposed two domain-independent graph representations to cover text, web-pages, and email: i) *Tree*, and ii) *Star*. Tree representation starts with the type of document as the root and then branches out based on the domain. The representation of an email message under this representation is shown in Figure 2.

This scheme considers all the information in an email message with each word in the email connected to the central root vertex. Figure 3 illustrates the *star representation*. It consists of a central anchor or root vertex. The chosen words from the email form the remaining vertices, along with the edges that connect them to the central root vertex with the edge *contains*. The ability to label edges makes this simple representation quite effective if the labels corresponds to the various components of a document, email or web page. Other representations are possible and the uniqueness of the proposed approach is that the process is independent of the representation. The accuracy of the results is dependent on the representation and the appropriateness of the representation to the domain under study.

It is easy to observe that the tree representation is richer than the star representation. The richer the representation, the better the classification accuracy is likely to be. Star has been proposed as it comes close to earlier approaches such as n-gram. Our hypothesis about the superior performance

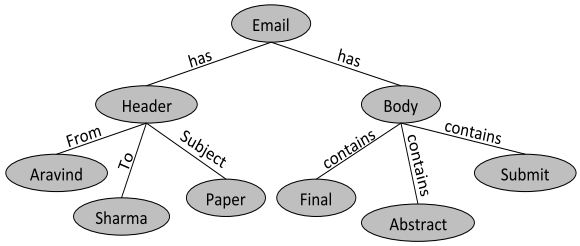

Figure 2. Email: Tree Representation

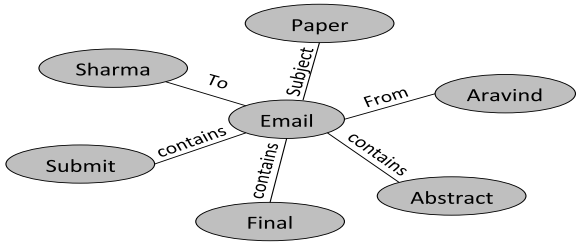

Figure 3. Email: Star Representation

of tree representation is confirmed when compared with both star and Naive Bayes representations.

IV. Substructure Extraction, Pruning and Inferring Parameters

As mentioned above, we extract interesting representative substructures (i.e., connected subgraphs within a given graph). Subdue is used for this purpose which uses an information-theoretic model for determining the best substructure given a forest of unconstrained graphs. The input to Subdue is a forest of graphs and the output is a set of substructures that are ranked based on their ability to compress the input graph using the *Minimum Description Length* [22] (MDL) principle. Subdue generates the best substructures that compress the input graph the most and lists out the *top-n* substructures.

In order to increase the effectiveness of Subdue for the task of email classification, we have to choose a number of input parameters that determine the number and type of substructures identified during substructure discovery. The best way to do this is to use the training set itself in order to derive these parameters. Certain characteristics of the class need to be taken into account in order to determine the representative substructures that best characterize a particular class. These parameters must be tunable and effective for diverse document and class characteristics. Moreover, not all classes exhibit similar properties; certain classes may be more dense as compared to others and certain others may have larger document content providing extensive amounts of information for training the classifier. For instance, in the email domain, due to constant addition, deletion and movement of emails, the folder contents keep changing rapidly. Hence, class 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 salient traits of the class. If the discovery process is guided by these parameters, the substructures generated are likely to better reflect the contents of the class.

An important aspect of Subdue's discovery process, *inexact graph discovery* [24], is important for our classification approach. Subdue groups similar substructures as a single substructure for both identification and representation. The distortion between two substructures might be a variation in the edge or in the vertex descriptions such as an addition, deletion or substitution of vertices or edges. Two substructures are considered to be isomorphic as long as the cost difference in generating both the substructures to be identical falls within the range the user considers acceptable. However, finding similar substructures is an NP-complete problem and requires an exponential algorithm. Subdue uses a *branch-and-bound* algorithm that is executed in polynomial time by considering reduced number of mappings.

This process of inexact graph match is vital for *m-InfoSift*, since it allows for substructures that vary slightly in their vertex or edge or label descriptions to be chosen as instances of a single substructure. The amount of variation permissible is determined by the *threshold* parameter input to Subdue. It specifies the bound on the difference that is allowed between instances of a substructure. The actual number is determined by the formula given in (1).

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

A small value of *threshold* allows a significant amount of inexactness while comparing substructure instances of documents that contain a large number of words. It is because even with a small value, the value computed by Equation 2 would allow reasonable number of variations. However, for documents with relatively smaller content and hence fewer vertices in the input graph representation, a larger value of threshold is required. The value of threshold needs to be determined based on the size of emails in a folder. If the amount of inexactness to be allowed in terms of the number of edge/vertex label variations is '*i*', then value of threshold is computed using Equation (2)

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

where, avg_s is the average size of the class documents.

The above formula adjusts the value of threshold taking into account the size of the documents in the class. For smaller documents, the value of the threshold is larger compared to the ones with larger sizes.

A. Substructure Pruning

The output of the discovery process generates a large number of substructures that contain minor variations of the same graph, and hence not all of them contribute towards the classification process. Furthermore, the cost of retaining and processing all the generated substructures is high since

it increases the overall time required for classification. Thus, pruning those substructures that do not influence the classification is desirable and is explained below:

Substructures with same frequency, size and MDL value: Substructures differing in either frequency, size or MDL value are retained to ensure uniqueness. For example, two substructures, each having ten vertices and different occurrence frequency, are not similar since the same substructure is not reported twice with different occurrence frequency. But two substructures, each with ten vertices and same occurrence frequency with minimal difference, such as different vertex or edge label, are pruned. Thus, each substructure in the representative set refers to an unique pattern that follows from the documents of the class under construction.

Substructures with low frequency in large classes: Using compression as a heuristic, the discovery algorithm identifies certain large substructures that do not occur frequently. It is due to the fact that replacing these huge substructures greatly compresses the original input graph. Hence, these substructures are returned by Subdue as part of the best substructure list even though they do not occur frequently. These substructures do not significantly add to the substructure set as they do not cover substantial portion of the class contents. Thus, substructures with very low frequencies as compared to the class size are discarded from consideration. The representative substructures generated from different categories (folders) are generated and pruned to retain unique substructures for ranking. We use two Subdue parameters to aid the pruning process – *NSubs* (that limits the number of substructures returned by Subdue) and *Beam* (that determines the number of best substructures retained at the end of each iteration of the discovery algorithm).

V. Substructure Ranking and Classification

A. Substructure Ranking

Once all the substructures from different classes have been generated and pruned, they are merged into a single list in order to be ranked. Some representative substructures occur more frequently or measure well in terms of size; hence, these can be considered to be more important than the others. It is therefore important to discriminate the substructures from the view point of classification. The earlier *InfoSift* framework currently ranks the substructures from each from in relation to the other substructures in the same folder. Extending this scheme for multiple folders/categories poses a problem of choosing the order of folder selection to be considered for classification as the test document can find a match in multiple representative substructures in different folders. A match with a higher ranked substructure in one category holds more weight than a match with a lower ranked substructure in another folder. The degree of match for a test document with the substructure is not represented by the rank of the substructure in the folder. For example, a test document can match with a representative substructure RS_1 ranked 10 belonging to Folder f_1 and also match with another representative substructure RS_2 ranked 1 belonging to folder f_2. Though it would be right to label the test document as belonging to folder f_2, the test document could match RS_1 with a better similarity therefore belonging to f_1. This calls for a scheme that could rank the substructures from each other based on their representativeness of their folder. The ranking formula and the intuition behind it explained below.

The pruned representative substructures list of each folder is merged into a single list to rank them globally. A rank is associated to each of the substructures based on the formula given in Equation 3. This formula is an extension and generalization of the TF-IDF formula to make it applicable to graphs instead of words. Based on this equation, the rank of the representative substructure *RS* is computed globally across all the folders in the training set. The rank comprises of the following characteristics of the substructure:

$$Rank(RS) = \left[\frac{FRS(f, RS)}{FRS(A, RS)} \times \frac{1}{IFF(RS)^2}\right] \times \frac{S_{RS}}{Max_{RS}} \tag{3}$$

where RS – Representative Substrucutre
$FRS(f, RS)$ – Frequency Term of RS across folder f
$FRS(A, RS)$ – Frequency Term of RS in Training set A
$IFF(RS)$ – Inverse Folder Frequency of RS
S_{RS} – Size of RS
Max_{RS} – Size of the largest RS in Global list

Frequency of Representative Substructure: Representative substructures are a group of words along with their occurrence in the structure throughout the document class. It is relevant to calculate their occurrence frequency in order to rank them in comparison with each other. The frequency term is shown in the enclosed square brackets in the Formula 3. It is based on the principle that the weight assigned to the representative substructure is proportional to its frequency in the folder it was extracted from and inversely proportional to its frequency across other folders. The frequency term comprises of three elements, thus, elaborating the importance of the substructure based on its occurrence frequency.

Frequency of Representative Substructure in A Folder: This term ($FRS(f, RS)$) captures the importance of the group of words that relate together in the representative substructure RS. The corresponding value of this terms is in turn computed by equation given in 4:

$$FRS(f, RS) = \frac{freq(RS, f_{RS})}{\sum_{i=0}^{n} freq(RS_i, f_{RS})} \tag{4}$$

where f_{RS} – folder for RS generation
$freq(RS, f_{RS})$ – frequency occurrence of RS in f_{RS}
$freq(RS_i, f_{RS})$ – frequency occurrence of ith RS in f_{RS}
n – total number of substructures within f_{RS}

The frequency of RS in f is given by Subdue in its output of best substructures, and can be computed by summing up the positive instances of all the representative substructures that have been extracted in folder f. This term has a *high value* when the representative substructure occurs more frequently across the document class it was generated from.

Frequency Term of Representative Substructure in All Folders: This term ($FRS(A, RS)$) represents the commonality of the representative substructure by computing the occurrence of RS through out the training set. The importance of this representative substructure RS is inversely proportional to its occurrence in other document classes since it does not aid in classification of test documents. This term can be computed by the equation 5:

$$FRS(A, RS) = \frac{\left[\sum_{j=0}^{m} freq(RS, f_j)\right] - freq(RS, f_{RS})}{\sum_{j=0}^{m}\sum_{i=0}^{n} freq(RS_i, f_j)} \quad (5)$$

f_{RS} – folder for RS generation
$freq(RS, f_j)$ – frequency occurrence of RS in folder f_j
$freq(RS_i, f_j)$ – frequency of RS_i in folder f_j
m – total folders in the training set
n – total number of substructures in folder f_j

The numerator of this term denotes the frequency of representative substructure RS in all the folders in the training set except the one it was generated in. This is computed by calculating the frequency of RS across all the folders in the training set and then discounting its occurrence frequency in the folder it was generated from. Checking whether RS exists in all other folders is not straight forward because RS might be a part of the representative substructures in other folders or the same words constituting RS might not be in the same order in the representative substructures of other folders. Inspecting whether a representative substructure occurs, as a whole or part of another substructure, in other folders is done using the *graph match* module of Subdue. The graph match module takes two graphs as input and computes the cost of transforming the largest of the input graphs into the smaller graph. The cost is computed by summing up the number of operations to be done on the larger graph, such as adding or deleting a vertex label or an edge label, to map it to the larger graph. The output of the graph match module comprises of mapping of largest graph to the smaller graph along with the cost.

The graph match module is used to examine whether representative substructure RS, that has been generated in folder f_{RS}, exists in other folders too. The representative substructures in other folders are considered to be similar to RS even if they contain RS as a subgraph in them. The algorithm shown in Figure 4 has been developed to find if two representative substructures are similar.

Determining Substructure Similarity

Step 1 : Obtain two representative substructures - RS1 and RS2.

Step 2 : Find largest substructure and initialize it
larger_one = RS2; smaller_one = RS1.

Step 3 : Compute difference in the size between the two:
(v2 –v1) + (e2 – e1)
where
v2 and e2 - number of vertices and edges of RS2
v1 and e1 - number of vertices and edges of RS1.

Step 4 : Compute the match cost between RS1 and RS2 using Graph-Match module.
Let M.C be the cost of transforming RS2 to RS1.

Step 5 : If **(M.C <= ((v2 –v1) + (e2-e1)))**
RS1 and RS2 are **similar** substructures
else
RS1 and RS2 are **not similar** substructures

Figure 4. Substructure Similarity Algorithm

The algorithm is initiated by finding the larger of the two subgraphs/substructures given as input. The largest subgraph is mapped to the smaller one as explained before. The difference between the sizes of the two substructures are found by the formula $(v2-v1)+(e2-e1)$. The match cost of transforming one substructure to another is computed using the graph match module. The difference in size is compared with the match cost computed. If the match cost is lesser or equal to the difference in size of the two substructure, then the two substructures are considered to be similar as both consist of the same vertex labels and edge labels. If the match cost is higher than the difference in the size of the two substructures, then the two substructures are not similar as labels of the vertices or edges are not the same.

Using this algorithm, substructures similar to the one under consideration are generated and correspondingly, the frequency of the substructure is computed by summing up the frequencies of all the similar substructures in other folders. The denominator of the term $FRS(A, RS)$, as shown in equation 5, is the summation of the frequencies of all the representative substructures in all the folders of the training set. This term determines the common representative substructure RS occurs across all the folders. The importance of RS is inversely proportional to its commonality in occurrence across folders.

Inverse Folder Frequency: This term determines the number of folders in which representative substructure RS occurs. The intuition is that, if a representative substructure occurs in many document classes, then it is not a good discriminator and it should be ranked lesser than the ones which occur in fewer document classes. This measure is computed while

calculating $FRS(A, RS)$ by maintaining an index of all the folders which contains representative substructures similar to RS. The IFF term provides a high value for common representative substructures and low value for unique ones (squaring gives a better non-linear effect). So when the inverse of IFF(RS) is considered, it provides a high value for rarely occurring substructure and vice versa.

Size of the Representative Substructure: The final term in the global rank formula in Equation 3 is the *size of the representative substructure*. Relatively large sized frequent substructures signify greater similarity among the documents in a class. The size of the representative substructure is compared with the largest substructure in the global list. Based on its relative size, a weight is evaluated to the representative substructure. Therefore, a representative substructure that compares well with the size of the largest substructure, is assigned a higher weight when compared to smaller substructures. A representative substructure gets a higher rank as – i) It occurs frequently across the same folder it was generated from, ii) It occurs less frequently across all the folders in the training set except the one it was generated from, and iii) The size of the representative substructure compares well to the largest substructure in the global list. Once all the representative substructure are ranked, they are ordered based on their ranks. The list of ordered substructures are then used during classification.

B. Classification

The ranked substructures are used for classifying the incoming documents. In order to assign an appropriate label to this document, it is compared with the ranked substructures. As with the generation of representative substructures, inexact graph match is used for comparing the unknown document with predefined representative substructures. Each substructure is included with the test document to create a forest of two graphs: 1) the test document represented as a graph, and 2) the graph of the representative substructure.

The classifier is used to generate representative substructures from the forest of graphs with a size set to the size of the representative substructure embedded in the test document. The list of substructures generated by the classifier are checked for its occurrence frequency. If the occurrence of the substructure is greater than 1, the test document has got an instance of the representative substructure in it. It also means that the test document comprises of the frequent words that represent the document class in the form of the representative substructure. Hence, the test document is filed to the class with the highest ranked substructure match signifying higher correlation with the class contents.

VI. Experimental Evaluation

A. Settings

Experimentally, it is somewhat difficult to compare the earlier approaches with ours as most of the email classification work is either rule-based or guided by the user interactively. In addition, as most of these systems are not available in open source, we have chosen the Naive-Bayesian approach for our comparison. As elaborated above, the entire operation of the framework is based on various parameters (*choice of graph*, *representation scheme*, *randomized generation*, *substructure discovery threshold*, etc.) associated with the different stages of the substructure discovery and classification process. These parameters are issued by the users in the form of a configuration file. We elaborate some of the crucial parameters and their values, that play an important role in the discovery and classification process:

Training-Test Set Split: The system has been evaluated using two different sets of ratios between the training and testing sets. For one set of experiments, we use a 80:20 split (i.e., 80% of the class documents represent the training set, and 20% represent the actual testing set). For the other batch of experiments, we use a test/train ratio of 60:40.

Seed: In case of random selection of documents for the training and test sets, a seed value can be provided for the randomized generation. For our set of experiments, we used the default value of 100.

Substructure Discovery Threshold: The amount of inexactness permissible during substructure discovery process is specified using this parameter. This value is calculated as explained in Equation 2. A value of 0.1 has been used for our experiments to compare the effect of inexact graph match on classification with exact graph match.

Classification Threshold: This value represents the amount of inexactness that is allowed to the classifier for grouping similar instances of patterns just as in substructure discovery. This value can be specified by the user else the same value as substructure discovery threshold is used as default. For our experiments, we have used a value of 0.05.

Beam: It represents the value of the *beam* for the substructure discovery algorithm. Values of the 2,4,8 and 12 have been used for the experiments. If the values are unspecified, then a value of 4 is used for small classes and a value of 8 has been used for medium and large classes by default.

The data-set for the experiments comprises of various folders that were selected from public Listserv's and personal emails from different persons in order to provide a diversity. The size of these folders varies from 10 to 470 emails. Additionally, to show that our approach is consistent and complete, experiments have also been carried on Enron Email Dataset [25] It comprises of data in the form of emails from about 150 users organized into multiple diverse folders. The experiments have been carried out on Intel Xeon CPU 2.80Ghz dual processor machines with 2GB memory. Exhaustive experiments on a large number of classes with diverse characteristics (different document class size, dense,sparse classes, etc.) have been carried on to study the effect of parameters on classification of unknown test documents in a multiple folder environment.

B. Results

1) Graph Mining vs. Naive Bayes: Figure 5 compares the performance of the *m-InfoSift* approach with the Probabilistic Bayesian approach. The results clearly show a significant improvement as compared to Bayesian. Accuracy improvement is 10% at the lowest and 70% at the highest. Naive Bayes assigns probabilities to word occurrences.

Terms that are common to multiple folders and having a higher weight assignment in one folder will outweigh others during classification which in turn leads to lot many wrongly classified emails. In other words, the classifier depends largely on the size of the classes in the training set. When a large class is paired up with small classes in the training data set, the probability that the most commonly occurring word in the test document to occur in the large class is higher and hence test documents are classified to the wrong folder.

Our approach, on the other hand, identifies patterns of word occurrences and rank them across all the folders thereby avoiding this problem. Additionally, the Naive Bayesian classifier fairs poorly due to large number of false positives. One of the main feature of *m-InfoSift* global rank scheme is the presence of fewer false positives (Figure 5).

The performance of *m-InfoSift* is consistently better across the Enron data-set as well. Though the difference in performance of both the classifiers is not as distinguishable as compared to the one from the Listserv dataset, our approach consistently outperforms the Naive Bayes approach. The results for the same are shown in Figure 6.

2) Effect of Feature-Set Size: The features that comprise the email graphs are chosen from the top *f%* of the sum of frequencies in folders making up the training data-set. Experiments have been carried out with three different values for retaining the frequent set of words: 60%, 80% and 100%. The results are shown in Figures 5 and 8.

When a value of 80% is used as the feature selection ratio, the performance of the classifier is consistently better as compared to retaining all words (100%), or a lower number of words (60%). The classification accuracy reduces with the increase in the number of folders. This behavior is expected as the chances of email getting correctly classified to the right folder decreases with the increase in the number of substructures in the global rank list.

3) Exact Vs. Inexact Graph Match: The ability to match similar substructure instances while making allowances for small variations is important when exact matches are hard to find. To validate this, we performed experiments to study the classification accuracy using *exact* and *inexact* graph match. As can be observed in Figures 7 and 8, *inexact* graph match exhibits better performance when compared to *exact* graph match. Emails do not correspond to a set of vocabulary and the information content of emails is sparse as compared to text documents. Therefore, it is difficult to find exact patterns throughout the document class and the ability to match instances with slight variations becomes significant.

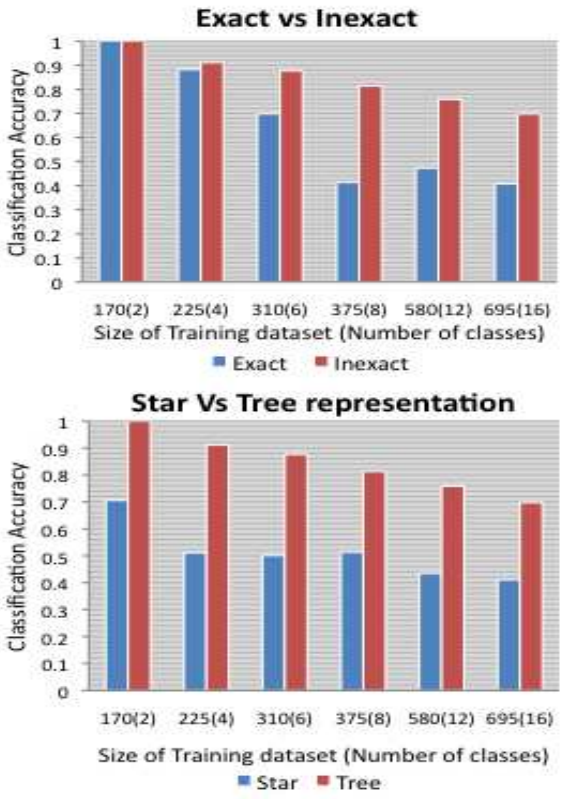

Figure 7. Substructure Discovery & Representations: Listserv dataset

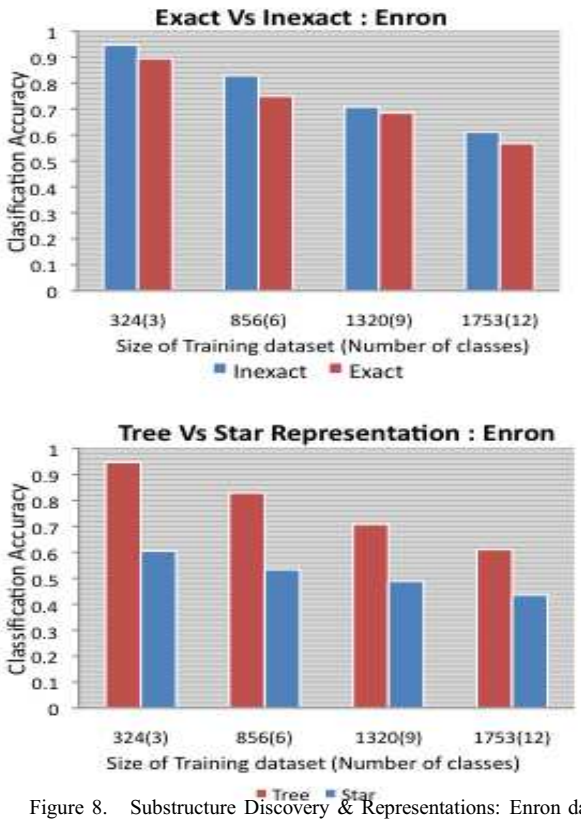

Figure 8. Substructure Discovery & Representations: Enron dataset

Figure 5. Naive Bayes vs. m-InfoSift - Listserv dataset

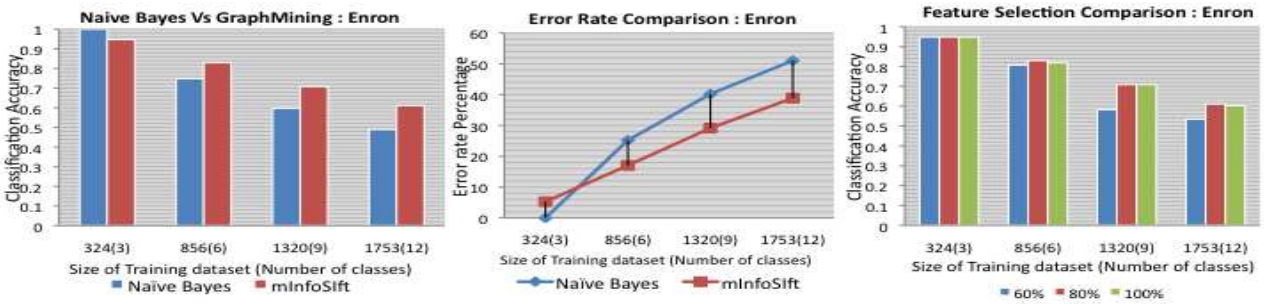

Figure 6. Naive Bayes vs. m-InfoSift - Enron dataset

4) Tree Vs. Star Representations: Experiments were conducted to study the effect of different graph representations on email classification. The *tree* representation showed better performance in classification as shown in Figure 7 for the Listserv dataset. The difference in performance between *star* and *tree* representation is attributed to the enhanced structural information captured by the tree representation. This is further reinforced with the results from the Enron data set (Figure 8).

C. Observations

Based on the results of various types of experiments carried out, we draw the following conclusions on the problem of email classification:

i) The performance of the system is consistent though the classification accuracy reduces with the increase with the training and test data size, which is expected. This aspect can be used to our advantage by using the latest emails in the folder for training instead of all of them. This also indicates the changing nature of folder characteristics with time.

ii) Emails, although comprising of unstructured text, have a structure and can be exploited for classification purposes.

iii) A structure that maps the document better intuitively is likely to improve the accuracy of classification as we have see in tree versus the star representations.

iv) The performance of the *m-InfoSift* for multiple folders is significantly better than Naive Bayes and is consistent across different categories of emails.

v) It is extremely important to derive **all** parameters for graph mining and classification from the folder itself which insures adaptability of the approach to arbitrary folders.

VII. Conclusions

We introduced a framework for effectively classifying emails into using graph mining techniques. The crux of the problem was to extract representative substructures and rank them. We have proposed a ranking formula for ordering the representative substructures generated from each document class in the training set. The characteristics as well ranking proposed in this paper is adaptive as it adjust to the size and characteristics of a folder. Various parameters that affect the ranking, and therefore classification, have been identified and analyzed in detail. The concept of feature subset selection enables us to classify data even when the amount of data available for training purpose is small. Alternative graph representations (such as tree and star) have been proposed in order to represent the documents and to incorporate useful and relevant domain information. The experimental results validate our framework by showing significant performance improvement over Naive Bayesian approach for varied email drawn from different domains.

Currently, we are investigating incremental generation of

representative substructures as the folders change over a period of time. We are also investigating how representative substructures change over a period of time and whether that information can be used to develop heuristics/rules to describe the manual classification process.

REFERENCES

[1] 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*, pp. 107–112, 2000.

[2] J. D. M.Rennie, "ifile:an application of machine learning to e-mail filtering," *Proceedings of KDD-2000 Text Mining Workshop,Boston Aug*, 2000.

[3] W. Cohen, "Learning to classify English text with ILP methods," in *Advances in Inductive Logic Programming*, L. De Raedt, Ed. IOS Press, 1996, pp. 124–143. [Online]. Available: citeseer.ist.psu.edu/cohen96learning.html

[4] A. K. McCallum and K. Nigam, "A comparison of event models for naive bayes text classification," *In the 15th Annual Internation ACM SIGIR Conference on Research and Development of Information Retrieval (SIGIR'92)*, pp. 59–64, 1992.

[5] T. Joachims, "Text categorization with support vector machines: Learning with many relevant features," *ECML*, pp. 137–142, 1998.

[6] C. Apte, F. Damerau, and S. M. Weiss, "Text mining with decision trees and decision rules," *Conference on Automated Learning and Discovery*, 1998.

[7] W. Lam and C. Ho, "Using a generalized instance set for automatic text categorization," *In the Proceedings of the 21st Annual Internation ACM SIGIR Conference on Research and Development of Information Retrieval (SIGIR'98)*, pp. 81–89, 1998.

[8] Y. Yang and C. G. Chute, "An example-based mapping method for text categorization and retrieval," *ACM Transactions on Information Systems (TOIS)*, vol. 12(3), pp. 252–277, 1994.

[9] C. Apte, F.Damerau, and S. Weiss, "Towards language independent automated learning of text categorization models," *In the Proceedings of the 17th Annual ACM/SIGIR conference*, 1994.

[10] H. T. Ng, W. B. Goh, and K. L. Low, "Feature selection, perceptron learning, and a usability case study for text categorization," in *Proceedings of SIGIR-97, 20th ACM International Conference on Research and Development in Information Retrieval*, N. J. Belkin, A. D. Narasimhalu, and P. Willett, Eds. Philadelphia, US: ACM Press, New York, US, 1997, pp. 67–73. [Online]. Available: citeseer.ist.psu.edu/ng97feature.html

[11] L. D. Baker and A. K. McCallum, "Distributional clustering of words for text categorization," *In Proceedings of the 21st Annual Internation ACM SIGIR Conference on Research and Development of Information Retrieval (SIGIR'98)*, pp. 96–103, 1998.

[12] M. Sahami, D. Heckerman, and E. Horovitz, "A bayesian approach to filtering junk e-mail," *AAAI-98 Workshop on Learning for Text Categorization*, 1998.

[13] G. Boone, "Concept Features in Re:Agent, an Intelligent Email Agent," in *Proceedings of the 2nd International Conference on Autonomous Agents (Agents'98)*, K. P. Sycara and M. Wooldridge, Eds. New York: ACM Press, 9–13, 1998, pp. 141–148. [Online]. Available: citeseer.ist.psu.edu/boone98concept.html

[14] R. Segal and J. O. Kephart, "MailCat: An Intelligent Assistant for Organizing E-Mail," in *AAAI/IAAI*, 1999, pp. 925–926.

[15] T. Payne and P. Edwards, "Interface agents that learn: An investigation of learning issues in a mail agent interface," *Applied Artificial Intelligence*, pp. 1–32, 1997.

[16] P. Clark and T. Niblett, "The cn2 induction algorithm," *Machine Learning*, pp. 261–283, 1989.

[17] E. Crawford, J. Kay, and E. McCreath, "Automatic induction of rules for e-mail classification," *Proceedings of the Sixth Australasian Document Computing Symposium, Coffs Harbour, Australia*, 2001.

[18] J. Heflman and C. Isbell, "Ishmail: Immediate Identification of important information, AT&T Labs." MIT Artificial Intelligence Laboratory, Tech. Rep., 1995.

[19] S. Kiritchenko, S. Matwin, and S. Abu-Hakima, "Email Classification with Temporal Features," in *Intelligent Information Systems*, 2004, pp. 523–533.

[20] S. A. Macskassy, A. A. Dayanik, and H. Hirsh, "EmailValet: Learning User Preferences for Wireless Email," in *IJCAI Workshop on Learning About Users and Machine Learning for Information Filtering*, 1999, pp. 925–926.

[21] M. Aery and S. Chakravarthy, "eMailSift: Email Classification Based on Structure and Content," in *ICDM*, 2005, pp. 18–25.

[22] D. J. Cook and L. B. Holder, "Substructure discovery using minimum description length and background knowledge," *Journal of Artificial Intelligence Research*, vol. 1, pp. 231–255, 1994. [Online]. Available: citeseer.ist.psu.edu/article/cook94substructure.html

[23] I. Guyon and A. Elisseeff, "An introduction to variable and feature selection." *Journal of Machine Learning Research*, vol. 3, pp. 1157–1182, 2003.

[24] H. Bunke and G. Allerman, "Inexact graph match for structural pattern recognition," *Pattern Recognition Letters*, pp. 245–253, 1983.

[25] http://www.cs.cmu.edu/ enron/, "Enron email dataset."