InfoSift: Adapting Graph Mining Techniques for Text Classification[*]

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

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. Various machine learning, information retrieval and probability based techniques have been proposed for text classification. In this paper we propose a novel, graph mining approach for text classification. Our approach is based onthe premise that representative – common and recurring –structures/patterns can be extracted from a pre-classified document class using graph mining techniques and the same can be used effectively for classifying unknown documents. 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 class contents. Extensive experimentation validate the selection of parameters and the effectiveness of our approach for text classification. We also compare the performance of our approach with the naive Bayesian classifier.

Introduction

Today, 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 automate information access based on relevance. One such mechanism is text classification, which allows a user 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 the same. There is reason to believe that documents within a class adhere to a set of patterns and that these patterns closely correspond to, and can be derived from the documents of that class. A classification system that determines the patterns of various term associations that emerge from documents of a class and uses these patterns for classifying similar documents is required. The ability to classify based on similar and not exact occurrences is singularly important in most classification tasks, as no two samples are exactly alike. In this paper, we propose a novel approach that adapts graph mining techniques for text classification. To the best of our knowledge, ours is the first attempt at applying graph mining techniques for classification (Aery & Chakravarthy 2004).

[*]This work is supported, in part, by NSF grants IIS-0097517 and IIS-0326505

Related Work

For text classification, a number of approaches have been proposed, these include Support Vector Machines (SVM) (Joachims 1998), Decision trees (Apte, Damerau, & Weiss 1998; Joachims 1998), k-Nearest-Neighbor (k-NN) classification (Lam & Ho 1998; Masand, Linoff, & Waltz 1992; Yang 1994), Linear Least Squares Fit technique (Yang & Chute 1994), rule induction (Apte, F.Damerau, & Weiss 1994; Cohen 1995; Moulinier, Raskinis, & Ganascia 1996), neural networks (Weiner, Pederson, & Weigend 1995; Ng, Goh, & Low 1997) and Bayesian probabilistic classification (McCallum & Nigam 1992; Baker & McCallum 1998; Koller & Sahami 1997; Tzeras & Hartman 1993). Support Vector Machines work by constructing a hyperplane that separates positive and negative examples of a class. A decision tree makes recursive splits based on discriminating attributes to determine the class of an unknown sample. k-NN techniques work by assigning the unknown document the class label corresponding to the majority of the k known documents it matches closely. While these techniques work well, they rely on extracting keywords or highly frequent words for classification and ignore the importance of a group of related words that co-occur. As stated before, documents of a class exhibit structures/patterns, which can be learnt and used for classifying unknown documents. Data mining is the process of discovering implicit and previously unknown patterns in data (Frawley, Piatetsky-Shapiro, & Matheus 1991). Therefore, we believe data mining techniques can be meaningfully applied to the task at hand and we have chosen graph data mining for our work as we intend to extract patterns occurrences instead of word occurrences. Graph models have been used to classify web documents (Schenker *et al.* 2003), but an extension of the *k-NN* algorithm is used to handle graph based data. The graph theoretical-distance measure for computing the distance translates to the maximal common subgraph distance proposed in (Bunke & Shearer 2001). A graph-encoded linguistic scheme has been applied for text classification in (Gee & Cook 2005).

Contribution

The main contribution of this paper is in the adaptation of a novel, but powerful approach, viz. graph mining for text classification and demonstrating its effectiveness. To the best of our knowledge, ours is the first attempt to assess the applicability of graph mining for classification. Another contribution is the identification of the graph mining approach (from among the alternatives) that is suited for classification. Graph mining (or any other mining technique) does not perform classification and hence its adaptation for classification is not straightforward. A number of parameters have been analyzed for our problem and their values have been identified both analytically and experimentally for performing classification. Finally, the last but not the least contribution of this paper is an extensive evaluation of the chosen graph mining technique and its comparison with a traditional classification technique.

Graph Based Data Mining

Graph mining, as opposed to transaction mining (association rules, decision trees and others) is suitable for mining structural data (chemical compounds, proteins, DNA, etc.) which have an inherent structure. The complex relationships that exist between entities that comprise these structures can be represented in graph format. The representation preserves the structural information of the original application which may be lost when the problem is translated to other representation schemes. Also, the various associations between objects in a complex structure are easy to understand and represent graphically. Documents and web pages have a structure in the form of the title, keywords, section headings, the HTML tag elements in case of web pages and the document body. By using the structural information present in the text, the inherent structure in the document is preserved, but graph mining approaches ar required for analysis. Relevant work in graph mining includes the Subdue substructure discovery system by Cook and Holder (Cook & Holder 2000). Subdue employs beam search to discover interesting and repetitive subgraphs, and compresses the original graph with instances of the discovered substructures. The frequent subgraphs approach by Kuramochi and Karypis (Kuramochi & Karypis 2001) maps the *apriori* algorithm to structural data represented in the form of a labeled graph and finds frequent itemsets that correspond to recurring subgraphs in the input. gSpan (Yan & Han 2002) uses a canonical representation by mapping each graph to a code and uses depth-first search to mine frequent subgraphs. Briefly, our work requires a means of substructure discovery directed by specific constraints (explained later). The notion of matching inexactly within bounds dictated by various domain characteristics is necessary. FSG and gSpan do not have this notion of matching inexactly within a threshold value as they use canonical labeling. Subdue satisfies our requirements of guided discovery and hence we have chosen the same for our work. For additional explanation, refer to (Aery Dec 2004).

Subdue Substructure Discovery System

Subdue (Cook & Holder 2000) is a graph based mining algorithm. It accepts a forest as input and identifies the best subgraph that minimizes the input graph using the minimum description length (MDL) principle (Rissanen 1989). It outputs subgraphs of different sizes and their occurrence frequency in the input graph. 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.

Inexact graph Match: A substructure is a connected subgraph within the graph representation. Though exact graph match comparison discovers interesting substructures, most of the substructures in the graph may be slight variations of another substructure. 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 (Bunke & Allerman 1983) is used where each distortion (addition, deletion or substitution of vertices or edges) is assigned a cost. The two graphs are said to be isomorphic as long as the cost difference falls within the user specified threshold.

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 to our work are discussed below:

1. ***Threshold***: The $threshold$ parameter provides a similarity measure for the inexact graph match. It specifies the amount of variation between instances of a substructure. The instances match if $matchcost(sub, inst) <= size(inst) * threshold$.
2. ***Nsubs***: This argument specifies the maximum number of the best substructures returned.
3. ***Beam***: 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.
4. ***Size***: This parameter is used to limit the size of the substructures that are considered. Size refers to the number of vertices in the substructure. A minimum and maximum value is specified that determines the range of the size parameter.

With this discussion of graph mining and Subdue, we will now discuss the adaptation of graph mining techniques for text classification.

InfoSift: Overview

The InfoSift system uses graph mining techniques to classify unknown documents into pre-classified document classes. The overall flow is as shown in Fig. 1. Each of the steps in the process are briefly described below:

Document pre-Processing: The pre-classified document classes form the training set for generating interesting and repetitive patterns. Stop word removal is peformed during document pre-processing and the same is carried out on the unknown document to eliminate commonly occurring words that carry no special meaning. Various characteristics of the class (e.g., the number of documents in the class, their size and so on) need to be taken into account in order to derive the optimal set of parameters for substructure discovery.

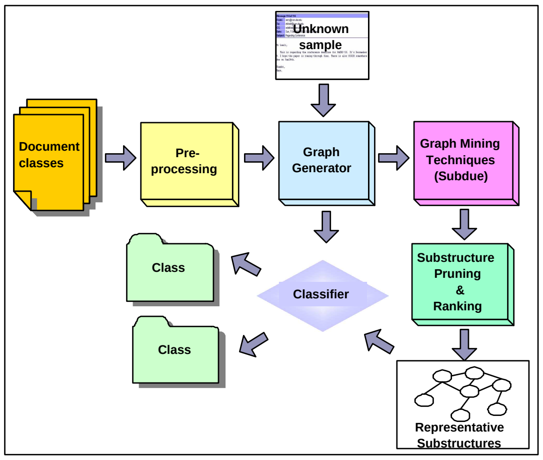
Unknown sample
Document classes
Pre-processing
Graph Generator
Graph Mining Techniques (Subdue)
Class
Classifier
Substructure Pruning & Ranking
Class
Representative Substructures

Figure 1: InfoSift Text Classification System

Graph Representation: The next step is to choose an appropriate graph representation and use it for representing documents in a class. We will discuss a canonical graph representation scheme that can be used for general text, emails and web pages.

Substructure Extraction: Graph mining techniques are used for extracting representative substructures. The substructure discovery process is driven by parameters derived from the characteristics of the class under consideration.

Representative Substructure Pruning: The output of the discovery process may contain a large number of substructures. Retaining all of them may not be necessary and the goal of pruning is to identify the subset needed for discriminating incoming unknown documents during classification.

Representative Substructure Ranking: It is important to discriminate among the set of pruned representatives 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 documents.

Classification: The representative substructures obtained and ranked for that class are compared with the unknown document to determine if any of them appear in the same. If a match is found, the document is classified to the corresponding category. For multi-way classification, in case of more than one match, the ranks associated with the matched substructures are used for resolving the class association. The document is classified into the category with the highest ranked substructure match.

Component Details & Discussion of Parameters

This section presents a detailed discussion of each step in the classification process. The influence of class characteristics and their mapping into parameters that affect substructure discovery along with the rationale for the choice of parameters and alternatives considered are discussed.

Document Pre-processing: 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 why, and what need to be retained. Before generating the input graph for a class, the documents in the same are processed to eliminate stop words. Since the goal is to retain substructures that are frequent across documents in a given category or class, the terms that comprise the substructures have to be frequent as well. These terms must occur frequently, preferably across a large percentage of the documents in the class. This choice of retaining words that are common across documents takes care of the disparity in document size, as some are considerably larger than others. Words are ranked based on their occurrence frequency across all documents in a class and those whose frequencies account for more than f% of the sum of all frequencies are retained. 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. The documents in the class are pre-processed as explained above and converted into a graph (actually a forest) for substructure discovery.

Graph Representation: The first step in discovering representative patterns is to transform the documents into a graph format for mining. The choice of 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. As we believe that our current approach can be useful for classifying general text, emails and web pages, we have proposed a canonical graph representation that can be used for any type of text. The representation is a *star-graph* and consists of a central anchor or root vertex and the chosen words from the document form the remaining vertices. These are connected to the central vertex with edges that have descriptive edge labels such as $contains$. The ability to label edges makes this simple representation quite effective. Fig. 2 shows such a representation where the edge labels used are $title$ and $contains$ to differentiate between various parts of a document. The representation can be further refined to include keywords, section headings and so on.

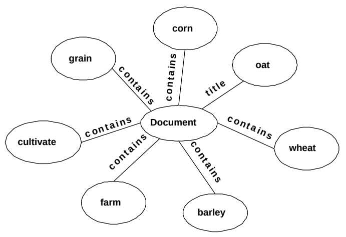

Figure 2: Canonical Graph Representation

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.

Impact of Class Characteristics

In order to identify representative structures of a given class, we have to choose a number of input parameters for the Subdue algorithm (such as $beam$, $threshold$, etc.). We believe that the document class itself needs to be used as a source for deriving these parameters as classes vary in their characteristics. Certain classes may be more dense as compared to others, certain others may have larger document content providing a greater amount of information for training the classifi er. These and other class characteristics need to be extracted during pre-processing and specifi ed as input parameters to the discovery algorithm to ensure that the substructure discovery process is based on the traits of the class. This way, the representative structures generated will be customized to a class and not based on a fi xed set of parameter values. We need to identify the parameters that are important and then provide a way for deriving them with substantiation. Below, we discuss the key characteristics of document classes that we believe affect substructure discovery and how they can be mapped as parameters for the same.

Average 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

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

Since each document has a different size (even after preprocessing), we use the average document size as one of the characteristics of a class. We argue that a low value of *threshold* allows for a signifi cant amount of inexactness while comparing substructure instances of documents that contain a greater a number of words. This is because, for larger documents, 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 identifi ed. For smaller documents and hence fewer vertices in the input graph representation, a larger value of threshold is required for inexact match that allows the same number of variations as in the former case. Using the size of the documents in a class, 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

$$threshold = \frac{i}{Avg_s} \qquad (2)$$

where, Avg_s is the average size of the documents in a class

Here, we have interpreted the average size as a parameter that affects pattern discovery and used it to derive a value for the threshold that allows for 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.

Average size & Class size Vs Number of Substructures: The number of substructures 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 class, the parameter has to be derived from the class. If the average size of the documents in a class is large, then the number of subgraphs is also large. Similarly, if there are a large number of documents in a class, there probably are a large number of substructure instances as well as substructures themselves. We have derived the number of substructures by using both the size of the class and the average document size along with weights to emphasize each factor as given by the following equation

$$nsubs = w_1 \times C_s + w_2 \times Avg_s,\ w_1 > w_2 \qquad (3)$$

where, C_s is the size of the class and
w_1 is the weighting factor applied to the same
Avg_s is the average size of the documents in the class and
w_2 is the weight applied to the average document size.

The particular numeric value for w_1 depends upon the size of the class under consideration. For the Reuters-21578 corpus () used for experimentation, we have discriminated classes into small (less than 60 documents), medium (61 to 200) and large (201 documents and above) depending upon their sizes. Due to space constraints the discussion for small classes is presented here. Since Subdue picks substructures based on their ability to compress the original graph, for a small class, large substructures despite their low frequencies are picked as best substructures, since abstracting even their few instances results in greater compression of the original graph. To ensure that comparatively smaller sized repetitive substructures are picked, a large value of $nsubs$ is required. Therefore taking into account the need for a large '$nsubs$' with small class size and scaling it with increase in average document size, we arrive at the following weights for w_1 and w_2

$$nsubs = 1.25 \times C_s + 0.50 \times Avg_s \qquad (4)$$

The weight w_2 is fi xed at 0.50 for all class sizes as the product term scales with increase in average document size. The values have been determined based on experimental observations. Similar weight measures specifi c to medium and large sized classes have also been derived by experimentation.

Beam size: *Beam* determines the number of best substructures retained at the end of each iteration of the discovery algorithm. The discussion regarding parameter 'nsubs' entailed ensuring the value chosen for the number of substructures is optimal to cover the possible set of substructures. While this is required, it is also important to ensure that the interesting substructures discovered during each iteration of the discovery algorithm are available for further consideration. This can be effected by selecting the right value of beam size. A low value of *beam* will result in loss of some interesting substructures as they do not get a chance to be evaluated in subsequent iterations of the discovery algorithm. Experiments employing *beam* sizes of 2, 4, 6, 8, 10 and 12 were performed on the experimental classes. Depending upon class size, in almost all cases $beam$ values of 12 worked well for large classes. For small and medium sized classes, beam values that returned good results were 2 and 4 respectively. A larger value of $beam$ for these classes only leads to increased computation time and extra processing in terms of pruning the unnecessary substructures.

Substructure size Vs. Minsize: It is important to retain substructures that are capable of discriminating among classes. Substructures that are common across classes do not help in the classification process as they provide no discriminating capability. The substructure size can be constrained above a minimum size to pick substructures that contain something more than a common 'core'. It makes sense to constrain substructures above a certain size, as a match with a very small substructure is not indicative of similarity. In our representation scheme, the central 'document' vertex is present in all substructures and we have constrained the minimum size of substructures to be greater than 3 to accommodate documents that vary in size. The constraint is specified using the *minsize* parameter that is input to the Subdue system.

Substructure Pruning and Ranking: A large number of substructures are returned as part of the discovery process; however, they are all not interesting from the viewpoint of classification. Inexact graph match returns substructures similar to others in terms of occurrence frequency and substructure size, while varying in one or more vertex or edge descriptions. Retaining several structures which have the same frequency and size and vary slightly will not contribute to classification especially when inexact match is used. It is necessary to prune this set of substructures to retain those that are true representatives of a class, and to save space and reduce the computation during classification. To prune the possibly large set of substructures, only those substructures that differ in frequency and size are retained. Since the discovery algorithm uses compression as a heuristic it 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 class contents and are pruned. The representative substructures are then ranked based on the frequency of occurrence, average document size and substructure size. Substructures with sizes comparable to the average document size and which cover a reasonable number of documents are preferred, as this signifies correlation with class contents. The formula for rank takes into account the above and is computed by the following equation

$$R_s = \frac{S_s}{Avg_s} \cdot \frac{f_s}{f_L} \tag{5}$$

where, S_s is the size of the substructure
Avg_s is the average document size
f_s is the occurrence frequency of the substructure and
f_L is the frequency of the most repeated substructure.
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 documents in the same. Relatively large sized frequent substructures signify greater similarity among documents of a class. An incoming document that matches a substructure with a high value of rank in class C_1 as compared to an average ranked substructure in class C_2, is appropriately classified to C_1.

Classification: Each incoming document is compared with the set of ranked representative substructures (from high to low) of all classes. As with the generation of representative pattern subgraphs, inexact graph match is used for comparison. The document is filed to the corresponding class if a match occurs. In case of multiple matches, the document is filed to the class with the highest ranked substructure, signifying higher correlation with the class contents.

Experimental Evaluation

Implementation: The InfoSift system has been implemented in Perl. The input to the system consists of one or more classes and a number of parameters such as the split for cross validation and random or sequential choice for the split. The prototype pre-processes the classes, generates graphs, computes the various parameters and invokes the discovery algorithm, then prunes and ranks the output for classification. The outcome along with a number of other values generated are logged for further analysis. The experiments have been carried out on the benchmark Reuters-21578 corpus. The category distribution of the corpus is skewed with a majority of the categories containing few documents to a few containing thousands of documents. Also, the corpus contains unlabeled documents that have not been considered for the experimental analysis. The resulting set of over 13000 documents corresponding to 60 topic categories has been used for training and testing purposes. We have performed extensive experiments using a training and test set split of 80:20. The metric used for evaluation is accuracy, which is defined as the ratio of the number of correct category assignments to the number of documents to be classified. All experiments have been carried out on a Pentium Xeon 2.66 Ghz dual processor machine with 2GB memory.

InfoSift Vs. Naive Bayes: Extensive experiments were carried out to compare the performance of InfoSift with that of the probability based naive Bayesian classifier[1]. The 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 results of comparison with the naive

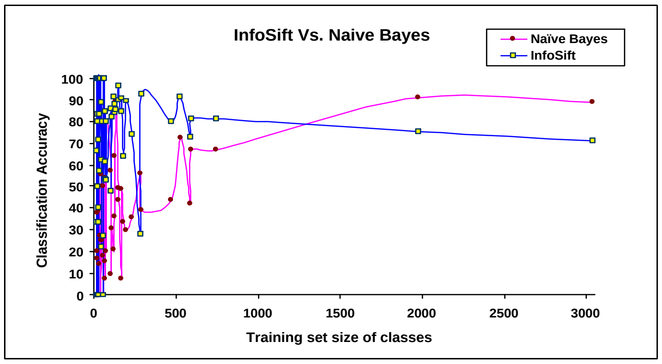

Figure 3: Performance of InfoSift Vs Naive Bayes

[1]As implemented in the Bow library by Andrew McCallum and publicly available at http://cs.cmu.edu/mccallum/bow

Bayesian probabilistic classifi er is presented in Fig. 3. The classifi cation accuracy of InfoSift is consistently better than the Bayesian classifi er, with the exception of the two large sized classes in the training set. The classifi er performs well for small and medium sized classes. A possible explanation for reduced accuracy on the large sets could be the relatively smaller beam value (of 12) used for experimentation. With a larger beam, a greater number of potential substructures can be given a chance for discovery. These will probably provide enough coverage for the contents of the class and lead to an increase in classifi cation accuracy.

Exact Graph Match Vs Inexact Graph Match: Experiments were performed to study classifi cation using exact and inexact graph match, as we believe the ability to match similar instances is important, especially in the textual domain, where exact matches are hard to fi nd. Our conjecture has been proved right by the experimental results. For smaller sized classes, it is clear that in the absence of large training data, inexact graph match does signifi cantly better than exact graph match as seen in the plot of Fig. 4. This is due to the fact that inexact graph match is able to group similar instances that vary slightly. This ability is more pronounced since the possibility of fi nding instances that match exactly is rare in smaller sized classes. For large classes, the classifi cation due to exact and inexact match is almost the same as seen in Fig. 5. Due to lack of space we are not able to

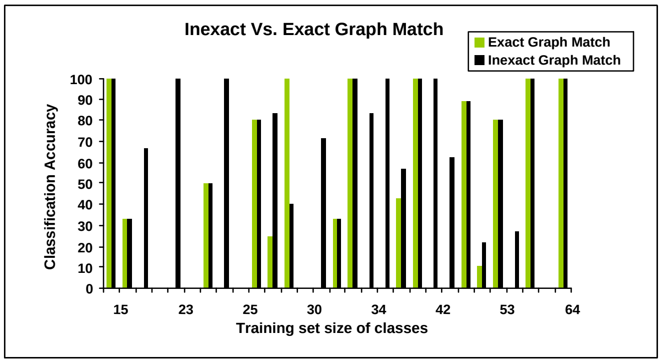

Figure 4: Exact Vs Inexact Match for Small Classes

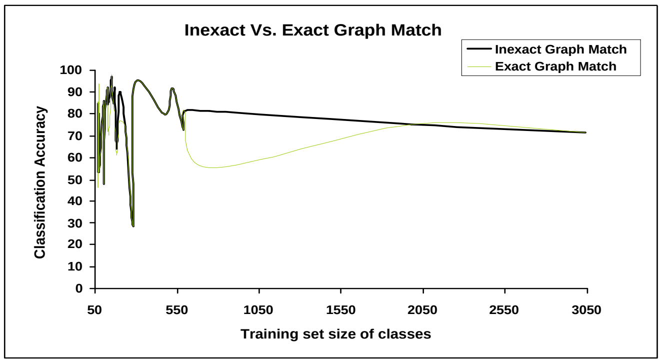

Figure 5: Exact Vs Inexact Match for Medium and Large Classes

present all of out experimental results. But the effectiveness of the classifi er can nevertheless be gauged from the experimental results that have been presented. For more details refer to (Aery Dec 2004).

Conclusions

In summary, we have proposed an innovative approach for document classifi cation using graph mining and the feasibility of the same for classifi cation has been established. We are able to make claims that our approach compares well with and even outperforms a conventional text classifi er. A detailed analysis of parameters that affect the classifi cation process has been presented and the experimental results validate the effectiveness of our approach.

References

Aery, M., and Chakravarthy, S. 2004. emailsift: Mining based approaches to email classifi cation. *In Proc. of the 27th ACM SIGIR Conf.*

Aery, M. Dec 2004. *InfoSift: Adapting Graph Mining Techniques for Document Classification*. Masters' Thesis, Dept. of Computer Science & Engg., Univ. of Texas at Arlington. http://itlab.uta.edu/ITLABWEB/Students/sharma/theses/Manu.pdf.

Apte, C.; Damerau, F.; and Weiss, S. M. 1998. Text mining with decision trees and decision rules. *Conf. on Automated Learning and Discovery*.

Apte, C.; F.Damerau; and Weiss, S. 1994. Towards language independent automated learning of text categorization models. *In Proc. of the 17th ACM SIGIR Conf.*

Baker, L. D., and McCallum, A. K. 1998. Distributional clustering of words for text categorization. *In Proc. of the 21st ACM SIGIR Conf.*

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

Bunke, H., and Shearer, K. 2001. A graph distance metric based on maximal common subgraph. *Pattern Recognition Letters* 753–758.

Cohen, W. W. 1995. Text categorization and relational learning. *In the 12th Intl. Conf. on Machine Learning (ICML'95).*

Cook, D. J., and Holder, L. B. 2000. Graph based data mining. *IEEE Intelligent Systems* 15(2):32–41.

Frawley, W. J.; Piatetsky-Shapiro, G.; and Matheus, C. J. 1991. Knowledge discovery in databases: An overview. *Knowlegde Discovery in Databases MIT Press* 1–27.

Gee, K. R., and Cook, D. J. 2005. Text classifi cation using graph-encoded lingusitic elements. *Proc. of the 18th Intl. FLAIRS Conf.*

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

Koller, D., and Sahami, M. 1997. Heirarchically classifying text using very few words. *In the 14th Intl. Conf. on Machine Learning (ICML'97)* 170–178.

Kuramochi, M., and Karypis, G. 2001. Frequent subgraph discovery. *IEEE Intl. Conf. on Data Mining* 313–320.

Lam, W., and Ho, C. 1998. Using a generalized instance set for automatic text categorization. *In Proc. of the 22nd ACM SIGIR Conf.* 81–89.

Masand, B.; Linoff, G.; and Waltz, D. 1992. Classifying news stories using memory based reasoning. *In Proc of the 15th ACM SIGIR Conf.* 59–64.

McCallum, A. K., and Nigam, K. 1992. A comparison of event models for naive bayes text classifi cation. *In Proc of the 15th ACM SIGIR Conf.* 59–64.

Moulinier, I.; Raskinis, G.; and Ganascia, J. 1996. Text categorization: a symbolic approach. *In the Proc. of the 5th Annual Sym. on Document Analysis and Information Retrieval*.

Ng, H. T.; Goh, W. B.; and Low, K. L. 1997. Feature selection, perceptron learning and a usability case study for text categorization. *Proc. of 20th ACM SIGIR Conf.*

Rissanen, J. 1989. Stochastic complexity in statistical enquiry. *World Publishing Company*.

Schenker, A.; Last, M.; Bunke, H.; and Kandel, A. 2003. Classifi cation of web documents using a graph model. *7th Intl. Conf. on Document Analysis and Recognition*.

Tzeras, K., and Hartman, S. 1993. Automatic indexing based on bayesian inference networks. *In Proc of the 16th ACM SIGIR Conf.* 22–34.

Weiner, E.; Pederson, J. O.; and Weigend, A. S. 1995. A neural network approach to topic spotting. *In the Proc. of the 4th Annual Sym. on Document Analysis and Information Retrieval*.

Yan, X., and Han, J. 2002. gspan:graph-based substructure pattern mining. *Proc. of the IEEE Intl. Conf. on Data Mining*.

Yang, Y., and Chute, C. G. 1994. An example-based mapping method for text categorization and retrieval. *ACM TOIS* 12(3):252–277.

Yang, Y. 1994. Expert network: Effective and effi cient learning from human decisions in text categorization and retrieval. *In Proc of the 17th ACM SIGIR Conf.* 13–22.