The Hybrid Representation Model for Web Document Classification, by A. Markov, M. Last, and A. Kandel, to appear in *Journal of Intelligent Systems*, Copyright © 2008 Wiley Periodicals, Inc.

The Hybrid Representation Model for Web Document Classification

A. Markov[1], M. Last[1♣] and A. Kandel[2]

[1]Department of Information Systems Engineering
Ben-Gurion University of the Negev
Beer-Sheva 84105, Israel
{markov,mlast}@bgu.ac.il

[2]Department of Computer Science and Engineering
University of South Florida
Tampa, FL 33620, USA
kandel@cse.usf.edu

Abstract. Most web content categorization methods are based on the vector-space model of information retrieval. One of the most important advantages of this representation model is that it can be used by both instance-based and model-based classifiers. However, this popular method of document representation does not capture important structural information, such as the order and proximity of word occurrence or the location of a word within the document. It also makes no use of the mark-up information that can easily be extracted from the web document HTML tags.

A recently developed graph-based web document representation model can preserve web document structural information. It was shown to outperform the traditional vector representation using the k-Nearest Neighbor (k-NN) classification algorithm. The problem, however, is that the eager (model-based) classifiers cannot work with this representation directly. In this paper, three new, hybrid approaches to web document classification are presented, built upon both graph and vector space representations, thus preserving the benefits and overcoming the limitations of each. The hybrid methods presented here are compared to vector-based models using the C4.5 decision-tree and the probabilistic Naïve Bayes classifiers on several benchmark web document collections.

The results demonstrate that the hybrid methods presented in this paper outperform, in most cases, existing approaches in terms of classification accuracy, and in addition, achieve a significant reduction in the classification time.

Keywords: Information retrieval, web content mining, document classification, graph theory.

1 INTRODUCTION

The huge amount of digital information stored on the web and on private intranets is growing at an amazing rate. Ever since the Internet and World Wide Web revolutionized

♣ Corresponding author

the information delivery technology, information overload[1] [16] has become a crucial problem in people's daily life. Under such circumstances, manual organization of documents is too costly and sometimes even impossible. Automated content-based document management methods, generally known as *information retrieval* techniques, are needed to deal with these amounts of data. Classification is one of the tasks involved.

Document classification (a.k.a *document categorization* or *topic spotting*) is the labeling of documents with a set of predefined thematic categories. The first document classification approaches belonged to the so-called *knowledge engineering* domain. These categorization techniques were based on rules separately generated by knowledge experts for each one of the available categories. Such rule generation was very expensive, and its prediction capability was low. Nowadays *machine learning* and *data mining* approaches are most commonly used for classification purposes. These techniques use a training set of pre-classified documents to build a classification model. This model is then used to classify previously unseen documents.

Web document classification became a very important sub-field of document categorization in the last decade due to the rapid growth of the Internet. Most web categorization methods originated in traditional text classification techniques that use only the HTML body text for document representation and classification model induction. Such an approach is not optimal for web documents, since it completely ignores the fact that web documents contain markup elements (HTML tags), which are an additional source of information. These tags can be used for identification of hyperlinks, the title, the underlined or bold text, etc. Furthermore, as demonstrated by Chakrabarti *et al.* in 8], a categorization algorithm can utilize the hyperlinks for exploring the "small neighborhoods" of web documents though this approach slows down the classification speed as it requires downloading the neighbor texts from the Internet. Major document representation techniques also give no weight to the order and position of words in the text. We believe that this kind of structural information may be critical for accurate web page classification. An enhanced document representation model is a solution for the problems explained above.

The Graph-Theoretic Web Document Representation Technique was recently developed [33]. The strength of the graph approach is in its ability to capture important structural information hidden in the document and its HTML tags. Capability to calculate the similarity between two graphs [5, 6] allows the classification of graphs using some distance-based lazy algorithms such as k-Nearest Neighbors (k-NN); the computational complexity of such algorithms is, however, very high. It is obvious, therefore, that lazy algorithms cannot be used for massive or online document classification. The major shortcoming of the graph representation is that most model-based classification algorithms, such as C4.5 [29], Naïve Bayes [15] and others cannot work with it. This fact prevents quick document categorization based on a pre-induced classification model.

In this paper we present a new method of web document representation, based on frequent sub-graph extraction that can help us to overcome problems of traditional bag-of-words [30] and graph techniques [4]. Our method has two main benefits: (1) we keep the important structural web page information by extracting relevant sub-graphs from a

[1] Information overload is defined as the problem that occurs when people are faced with so much information that they are unable to attend all of it.

graph that represents this page; (2) we can use most model-based classification algorithms for inducing a classification model because, eventually, a web document is represented by a simple vector with Boolean values.

2 Related Work

In information retrieval techniques, the vector space model [30] is typically used for document representation. A set of terms $T(t_1,...,t_{|T|})$ that occurred at least once in at least one document, serves as a feature set and each document d_j is represented as vector $\bar{d}_j = (w_1,...,w_{|T|})$, where each w_i is a significance weight of a term t_i in a document d_j. The set T is usually called vocabulary or dictionary[2]. The differences between the various approaches are in:

1. the method used to define a term
2. the method used to calculate the weight of each term

In traditional information retrieval techniques single words are used as terms. This method is called a 'set' or 'bag-of-words' and it is widely used in document categorization studies and applications. Some examples can be found in [25, 36, 21, and 7]. According to this approach, the vocabulary is constructed from either all or N most weighted words that appear in training set documents. Though this simple representation provides relatively good classification results in terms of accuracy, its limitations are obvious. This popular method of document representation does not capture important structural information, such as the order and proximity of term occurrence or the location of a term within a document.

As to the term weight calculation, the TF × IDF (term frequency × inverse document frequency) measure [31, 32] is most frequently used. Such a measure assigns the highest weight to terms that occur frequently in a specific document but do not occur at all in most other documents. The Boolean model of Information Retrieval is also very popular and it usually produces good accuracy results [1, 18]. Here a document is represented as a vector where dimension values are Boolean with 0 indicating the absence and 1 indicating the presence of the corresponding dictionary term in the document.

In a number of works more sophisticated text representation techniques were presented and evaluated. These representations did not yield better results compared to the bag-of-words in [31]. In [11], for instance, syntactic phrases were used as a supplement to single words for the improvement of text retrieval effectiveness. Syntactic phrase indexing is the use of syntactic analysis to produce multiword indexing terms. The phrasal term is considered to be assigned to a document only when all its component words appear in the document and have the proper syntactic relationship. Experiments presented in [11] have shown only minor improvements in some cases.

As shown by Hotho *et al.* in [17], the traditional bag-of-words text representation can be enhanced using background knowledge available in the form of ontology like Wordnet. The paper presents and evaluates several optional strategies for adding or replacing terms by concepts. The results of [17] are quite promising, especially when a domain-specific ontology is used. However, detailed and comprehensive ontologies are not available yet in many domains of human knowledge, while non-English lexicons for

[2] The difference between *dictionary* and *bag/set of words* is that in the first one, the *term* is not defined and can be any combination of characters and words, while in the second one, the *term* is a single word.

domain-specific ontologies are even scarcer.

The major goal of Lewis's experiment [20] was to compare four different ways to define terms: words, phrases, word clusters, and phrase clusters. Reuter's data set was used for comparison. Term clustering [20] is the use of cluster analysis in an attempt to group together semantically related indexing terms (classification features). Cluster analysis forms groups of objects which have similar values for some set of features. In term clustering, the objects to be clustered are indexing terms, which are themselves documents features. Lewis calls the features of indexing terms metafeatures[3]. Most research on term clustering has used metafeatures which correspond to documents. When clustering terms from a 200 document collection, each term would be represented by 200 metafeatures, with each metafeature indicating the presence or absence of that term in one of the 200 documents. Each cluster, in this case, will lead to terms that frequently occur together in the same documents. After clustering, all words in a specific cluster became a single term, so that, if some document includes all the words in the cluster, the weight of this term in the vector representation of this document will not be zero. In this particular work, metafeatures are the documents under the same category. The major conclusions of the research were:

1. Optimal effectiveness occurs when using only a small proportion of the indexing terms available;
2. Effectiveness peaks at a higher feature set size and lower effectiveness level for a syntactic phrase indexing than for word-based indexing;
3. Reported term clustering methods cannot provide an improvement in text representations compared to word term representation – bag of words.

Relational logic document representation that includes word order information is proposed in [9]. Labeled examples of the target class C are represented as facts of the form $+c(d)$ and $-c(d)$ for positive and negative examples, respectively. Here d is a constant that identifies a specific document. Documents are also identified with a set of facts of type $w_i\ (d, p)$, indicating that word w_i appears in the document d at position p. Positions are integers $1 \le p \le n$, where n is the length of the longest document. A set of facts from type $w_i\ (d, p)$ is used as background relationships or knowledge and makes it possible to define predicates needed for representation and categorization. Predefined predicates that were used are:

$near1(p_1, p_2)$ is true when $|p_1 - p_2| \le 1$
$near2(p_1, p_2)$ is true when $|p_1 - p_2| \le 2$
$near3(p_1, p_2)$ is true when $|p_1 - p_2| \le 3$
$after(p_1, p_2)$ is true when $p_1 < p_2$
$succ(p_1, p_2)$ is true when $p_2 = p_1 + 1$

Then c(d) facts are used as training examples, where each example is represented by a list of facts extracted from the predicates above. Classifiers that can learn data represented this way belong to the subfield of Machine Learning called Inductive Logic Programming or ILP. ILP and its learning principles are explained in greater detail in [12]. A popular Induction Rules Learning algorithm that can work with such a representation, and was used in [9], is FOIL [28]. An interesting implementation of the relational representation for web document classification is given in [10]. Its assumption is that taking into account relationships between pages in the representation stage can

[3] Metafeatures in this case are basically the documents themselves.

improve the classification performance. Some relational predicates that were used in order to make it possible are:

- *has_word(page)*: this set of Boolean relationship predicates indicates which word exists in which page. If the value of predicate is true then page *page* contains word *word*. A set of these predicates covers conventional bag-of-words encoding;
- *link_to (pageA, pageB)*: these predicates describe relationships between pages. If true then *pageA* points to *pageB* by hyperlink. The problem with this predicate is that, if *pageA* belongs to a training set and *pageB* does not, then the predicate is useless. In such a case, a hyperlink pointed to *pageB* should be ignored or *pageB* must be manually classified and added to the training set.

An Induction Rules Learning FOIL classifier was used by the authors for this specific research. The classification results obtained in the experiments described above were compared to the bag-of-words-based representations, using the Naïve Bayes classifier, on the same data set. The relational representation achieved a slightly higher level of accuracy. Since both the representation type and the classifier were replaced during the comparison, it is hard to say what caused this improvement: a better representation, a better classifier, or both.

The *n*-gram language models have also been used for various text classification tasks including authorship attribution, language identification, and topic detection (e.g., see [26] and [35]). An *n*-gram is simply a consecutive sequence of characters or words of a fixed window size *n*. The authors of [26] have enhanced the classical Naïve Bayes Classifier model by forming a Markov chain of consecutive attributes. They have experimented with various character and word level models where the order *n* was limited to the values of 8 and 4 respectively. However, on the topic detection task in a large collection of English documents (Reuter's Dataset), the absolute accuracy improvement vs. the state-of the-art text classification methods was quite marginal: at most 0.5% using the word level and at most 1.5% using the character level. The results of another study [35] have suggested that using bigrams *in addition to* unigrams can be helpful for more accurate identification of some document categories.

The graph-based approach to web document representation was introduced in [33]. Its overview is provided in Section 3 below. In [22], we have presented the "Naïve" method for term extraction from document graphs. The more sophisticated, "Smart" term extraction method was initially introduced by us in [23] and then enhanced in [24]. In this paper, we have significantly extended [24] by providing a comprehensive evaluation of the three proposed hybrid representations of web documents (Hybrid Naïve, Hybrid Smart, and Hybrid Smart with Fixed Threshold) using two model-based classifiers (C4.5 and Naïve Bayes) and four benchmark document collections.

3 Graph Document Models: An Overview

In this section, we describe a novel, graph-based methodology, designed especially for web document representation [33]. The main benefit of graph-based techniques is that they allow keeping the inherent structural information of the original document. Before describing the graph-based methodology, the definition of a graph, subgraph and graph isomorphism should be given.

A graph G is a 4-tuple: $G= (V, E, \alpha, \beta)$, where V is a set of nodes (vertices), $E \subseteq V \times V$ is

a set of edges connecting the nodes, $\alpha : V \rightarrow \sum v$ is a function labeling the nodes, and $\beta : V \times V \rightarrow \sum e$ is a function labeling the edges ($\sum v$ and $\sum e$ being the sets of labels that can appear on the nodes and edges, respectively). For brevity, we may refer to G as $G= (V, E)$ by omitting the labeling functions.

A graph $G1=(V_1,E_1,\alpha_1,\beta_1)$ is a subgraph of a graph $G_2=(V_2,E_2,\alpha_2,\beta_2)$, denoted $G_1 \subseteq G_2$, *if* $V_1 \subseteq V_2$, $E_1 \subseteq E_2 \cap (V_1 \times V_1)$, $\alpha_1(x) = \alpha_2(x) \ \forall x \in V_1$ and $\beta_1(x, y) = \beta_2(x, y) \ \forall (x, y) \in E_1$. Conversely, graph G_2 is also called a supergraph of G_1.

All graph representations proposed in [33] are based on the adjacency of terms in an HTML document. Under the *standard method*, each unique term (keyword) appearing in the document becomes a node in the graph representing that document. Distinct terms (stems, lemmas, etc.) can be identified by a stemming algorithm and other language-specific normalization techniques. Each node is labeled with the term it represents. The node labels in a document graph are unique, since a single node is created for each term even if a term appears more than once in the text. Second, if a word *a* immediately precedes a word *b* somewhere in a "section" *s* of the document, then there is a directed edge from the node corresponding to term *a* to the node corresponding to term *b* with an edge label *s*. An edge is not created between two words if they are separated by certain punctuation marks (such as periods). Sections defined for the standard representation are: title, which contains the text related to the document's title and any provided keywords (meta-data); link, which is the anchor text that appears in hyper-links on the document; and text, which comprises any of the visible text in the document (this includes hyperlinked text, but not the text in the document's title and keywords). Graph representations are language-independent: they can be applied to a normalized text in any language. An example of a standard graph representation of a short English web document is shown in Figure 1, where TL denotes the title section, L indicates a hyperlink, and TX stands for the visible text.

[Figure 1: Standard Graph Document Representation]

In principle, the graph-based document representations are equivalent to aggregations of word-level *n*-gram models, where the order *n* varies between 1 (unigrams) and the maximum size of a document graph. However, the methodology of [33] does not require or use the computation of probabilities for word sequences (chains) in a document.

After the representation stage, documents can be classified with the lazy k-NN classifier. Authors of [33] used several distance and similarity measures for classification. An example of such distance measure is given below.

$$d_{MMCSN}(G_1, G_2) = 1 - \frac{|mcs(G_1, G_2)|}{|MCS(G_1, G_2)|}$$

where:

- $d_{MMCSN}(G_1, G_2)$ - normalized distance between graphs G_1 and G_2 (*MMCSN* is for *M*aximum *M*inimum *C*ommon *S*ubgraph/*S*upergraph *N*ormalized)
- *mcs* (G_1, G_2) – maximal common subgraph of G_1 and G_2
- *MCS* (G_1, G_2) – minimal common supergraph of G_1 and G_2

The authors of [33] reported a significant improvement in classification accuracy achieved with graph vs. bag-of-words representation, using the k-NN classifier. In terms of time needed to classify one document, however, graphs were found to be much slower than vectors.

Another problem of graph representation is that documents represented by graphs cannot be classified with most model-based classifiers. On the other hand, the computational complexity of instance-based algorithms is typically very high and they cannot be used for massive online web document classification like the model-based data mining algorithms, which use the conventional vector-space representation and are generally much faster than the lazy ones.

4 Hybrid Document Models Using Graphs

4.1 Term Definition

In order to represent a web document, a term first has to be defined. The proposed methodology is based on graph document representation [33]. In the representation methods presented here, terms (discriminative features) are defined as subgraphs selected to represent a document already converted into a graph form. It is obvious that all possible subgraphs in a document graph cannot be taken as attributes because of their quantity, so some subgraph selection criteria and techniques are needed. In this work, three optional subgraph selection procedures are proposed, namely Hybrid Naïve, Hybrid Smart, and Hybrid Smart with Fixed Threshold.

At this point, it is important to emphasize that our methods are not aimed at identifying syntactic phrases (e.g., noun phrases, adjective phrases, etc.) in a document though some of extracted subgraphs may happen to be such phrases. Our goal is to select discriminative subgraphs in a given document *without* applying language-specific and computationally intensive operations like Part-of-Speech Tagging, Named Entity Recognition, and Sentence Parsing. Relying on either a general-purpose or a domain-specific ontology and a related lexicon (like in 17]) is also beyond the scope of this work.

4.2 Categorization Model Induction Based on a Hybrid Document Representation

The process for inducing a classification model from labeled web documents represented by graphs is shown in
Figure 2.

[Figure 2: Classification Model Induction]

First we obtain a training set of labeled *web documents* $D = (d_1, \ldots, d_{|D|})$ and a set of categories as $C = (c_1, \ldots, c_{|C|})$, where each document $d_i \in D;\ 1 \le i \le |D|$ belongs to one and only one category $c_v \in C;\ 1 \le v \le |C|$. Then graph representation of documents is generated (see Section 3) and a set of labeled graphs $G = (g_1, \ldots, g_{|D|})$ is obtained. Now we are able to extract predictive features (subgraphs in this case) by identifying the subgraphs, which are most relevant for classification in a set of training graphs. The Naïve or the Smart

methods can be used. A set of terms (subgraphs), or vocabulary $T = (t_1, ..., t_{|T|})$ is the output of this stage.

Using T we can now represent all document graphs as vectors of Boolean features for every subgraph term in the set T ("1" – a subgraph from the set, created in the previous stage, appears in the graph of a particular document; "0" - otherwise). *Feature selection* may be performed to identify best attributes (Boolean features) for classification. Then prediction model creation and extraction of classification rules can be performed by one of the "eager" classification algorithms. Naïve Bayes Classifier and the C4.5 algorithm were used for evaluation purposes in this particular research study.

4.3 The Hybrid Naïve Approach

The Naïve approach to term extraction was initially introduced by us in [22]. All graphs representing the web documents are divided into groups by class attribute value (for instance: *business* and *sports*). A frequent sub-graph extraction algorithm is then applied to each group with a user-specified threshold value t_{min}. Every subgraph more frequent than t_{min} is selected by the algorithm to be a term (discriminative feature), and stored in the vocabulary. All obtained groups of subgraphs (discriminative features) are combined into one set.

In this work, so-called the *Standard Graph Representation* from [33] was used. Standard representations and others proposed in [33], convert a document into a graph, where each node is labeled by the word it represents. An important property of the graphs is that the vertex labels are unique in each graph, which makes the graph and the subgraph isomorphism identification much easier than the standard subgraph discovery case [19, 37], where such a restriction does not exist. We used the FSG algorithm [19] for frequent subgraphs extraction with all selection methods.

The Naïve method is based on a simple postulation that a feature explains the category best if it appears frequently in that category; in real-world cases, however, this is not necessarily true. For example if a sub-graph g is frequent in more than one category, it can be chosen as a feature by the Naïve method though it is not helpful for making a distinction between documents belonging to those categories. The *Smart* extraction method presented in the next sub-section has been developed to overcome this problem.

4.4 The Hybrid Smart Approach

The first publication related to the Smart term extraction appears in [23]. As in the Naïve representation, all graphs representing the web documents are divided into groups by class attribute value. In order to extract subgraphs, which are relevant for classification, some measures are defined, as follows:

SCF – Sub-graph Class Frequency:

$$SCF\left(g'_k(c_i)\right) = \frac{g'_k f(c_i)}{N(c_i)}$$

Where

$SCF\left(g'_k(c_i)\right)$ - Frequency of sub-graph g'_k in category c_i .

$g'_k f(c_i)$ - Number of graphs containing a sub-graph g'_k in category c_i.
$N(c_i)$ - Number of graphs in category c_i .

ISF - Inverse Sub-graph Frequency:

$$ISF(g'_k(c_i)) = \begin{cases} \log_2\left(\frac{\sum N(c_j)}{\sum g'_k f(c_j)}\right) & \text{if } \sum g'_k f(c_j) > 0 \\ \log_2\left(2 \times \sum N(c_j)\right) & \text{if } \sum g'_k f(c_j) = 0 \end{cases} \qquad \{\forall c_j \in C;\ j \neq i\}$$

$ISF(g'_k(c_i))$ - Measure for inverse frequency of sub-graph g'_k in category c_i .
$N(c_j)$ - Number of graphs in category c_j.
$g'_k f(c_j)$ - Number of graphs containing g'_k in category c_j.

And finally we calculate the *CR – Classification Rate*:
$CR(g'_k(c_i)) = SCF(g'_k(c_i)) \times ISF(g'_k(c_i))$
$CR(g'_k(c_i))$ - Classification Rate of sub-graph g'_k in category c_i . The interpretation of this measure is how well g'_k explains category c_i . $CR(g'_k(c_i))$ reaches its maximum value when every graph in category c_i contains g'_k and graphs in other categories do not contain it at all.

According to the Smart method, CR_{min} (minimum classification rate) is defined by the user and only sub-graphs with CR value higher than CR_{min} are selected as terms and entered into the vocabulary. The calculation of the Classification Rate for each candidate subgraph is a slightly more complicated and time-consuming procedure in the *Smart* approach than finding only the subgraph frequency because of the *ISF* (Inverse Sub-graph Frequency) calculation where graphs from other categories are taken into account. Notwithstanding, as can be seen below, in some cases using Smart representation produces better results in terms of accuracy.

4.5 The Hybrid Smart Approach with Fixed Threshold

In this type of extraction we define a minimal classification rate CR_{min} together with the minimal frequency threshold t_{min}. In order to select a subgraph g'_k as relevant for classification, two conditions should be met:

- $SCF\ (g'_k(c_i)) > t_{min}$
- $CR\ (g'_k(c_i)) > CR_{min}$

The first condition was added because, in some cases, when a subgraph is infrequent in some category but even less frequent or non-existent in other categories it can still pass the CR_{min} threshold. This hypothesis is theoretically logical, but in practice it did not provide a significant improvement in classification accuracy. However, introduction of a fixed threshold for additional elimination of non relevant subgraphs should reduce the computation time. The extraction process is similar to the Smart extraction with one small difference – when Sub-graph Class Frequency *(SCF)* is calculated for a specific term, it is compared to t_{min}. If $SCF \leq t_{min}$ – the subgraph is dropped, otherwise we proceed with calculating the classification rate *CR*.

4.6 Frequent Sub-Graph Extraction

The input of the sub-graph discovery problem is, in our case, a set of labeled, directed graphs and threshold parameters t_{min} and/or CR_{min}. The goal of the frequent sub-graph discovery is to find all connected sub-graphs that satisfy the classification relevancy constraints defined above. Additional property of our graphs is that a labeled vertex is unique in each graph. This fact makes our problem much easier than the standard sub-graph discovery case [19] where such restriction does not exist. The most complex task in frequent sub-graph discovery problem is the *sub-graph isomorphism identification*[4]. It is known as NP-complete problem when nodes in the graph are not uniquely labeled but in our case it has a polynomial $O(n^2)$ complexity. We use *breadth first search* (BFS) approach and simplify the FSG algorithm given in [19] for sub-graph detection.
Our *Naïve* algorithm for frequent subgraph extraction and its notations are presented in Algorithm 1 and Table 1 respectively. First, all frequent nodes in the input set of graphs are detected and inserted into the frequent subgraph set. At each iteration of the *While* loop (Row 3), we try to extend each frequent subgraph of size *k* by finding subgraph isomorphism between it and the graphs from the input set and adding outgoing edge to the subgraph (Row 7). Then we construct a set C^k of all possible candidate subgraphs (Rows 8 to 13). We store frequent candidates in the frequent set F^k (Row 14) and return the union of all frequent subgraph sets obtained after each iteration (Row 16). The outline of the *Smart* approach to frequent subgraph extraction is given in Algorithm 2.

Table 1 Notations Used

Notation	Description
G	Set of document graphs
t_{min}	Subgraph frequency threshold
K	Number of edges in the graph
G	Single graph
sg	Single subgraph
sg^k	Subgraph with k edges
F^k	Set of frequent subgraphs with k edges
E^k	Set of extension subgraphs with k edges
C^k	Set of candidate subgraphs with k edges
CR_{min}	Minimum classification rate

[4] Means that a graph is isomorphic to a part of another graph.

```
Naïve-Extraction (G, t_min)
 1: F^0 ← Detect all frequent 1 node subgraphs (vertexes) in G
 2: k ← 1
 3: While F^(k-1) ≠ Ø Do
 4:     For Each subgraph sg^(k-1) ∈ F^(k-1) Do
 5:         For Each graph g ∈ G Do
 6:             If sg^(k-1) ⊆ g Then
 7:                 E^k ← Detect all possible k edge extensions of sg^(k-1) in g
 8:     For Each subgraph sg^k ∈ E^k Do
 9:         If sg^k already a member of C^k Then
10:             {sg^k ∈ C^k}.Count++
11:         Else
12:             sg^k.Count ← 1
13:             C^k ← sg^k
14:     F^k ← {sg^k in C^k | sg^k.Count > t_min * |G|}
15:     k++
16: Return F^1, F^2, ...F^(k-2)
```

Algorithm 1: The *Naïve* Approach to Frequent Subgraph Extraction

Smart-Extraction (G, $\bar{G}$, CR_{min})
1: $F^0 \leftarrow$ Detect all 1 node subgraphs sg^0 (vertexes) in G for which $CR\ (sg^0) > CR_{min}$
2: $k \leftarrow 1$
3: While $F^{k-1} \neq \emptyset$ Do
4: For Each subgraph $sg^{k-1} \in F^{k-1}$ Do
5: For Each graph $g \in G$ Do
6: If $sg^{k-1} \subseteq g$ Then
7: $E^k \leftarrow$ Detect all possible k edge extensions of sg^{k-1} in g
8: For Each subgraph $sg^k \in E^k$ Do
9: If sg^k already a member of C^k Then
10: $\{sg^k \in C^k\}.SCF += 1/|G|$
11: Else
12: $sg^k.\ SCF \leftarrow 1/|G|$
13: $C^k \leftarrow sg^k$
14: For Each $sg^k \in C^k$ Do
15: $sg^k.ISF \leftarrow$ ISF (sg^k, $\bar{G}$)
16: $sg^k.CR \leftarrow sg^k.\ SCF \times sg^k.ISF$
17: $F^k \leftarrow \{sg^k \in C^k \mid sg^k.CR > CR_{min}\}$
18: k++
19: Return $F^1, F^2, \ldots F^{k-2}$

Algorithm 2: The *Smart* Approach to Frequent Subgraph Extraction

4.7 Computational Complexity

The computational complexity of our algorithm is similar to the complexity of apriori based *FSG* [19] with little differences resulting from the way we are looking for subgraph extensions, that can probably be further optimized. Additional and much more important difference is in subgraph and graph isomorphism identification complexity, which are most time expensive tasks in the process of frequent subgraphs extraction. While graph isomorphism identification is not known to be either P or NP complete [13], subgraph isomorphism problem has been shown to be NP complete [14]. This, of course, relates to general graph's case when vertices are not uniquely labeled. In our case, graphs are directed and have unique vertices so subgraph isomorphism between graphs G_1 and G_2 ($G_1 \subseteq G_2$) can be determined by the following procedure:

1. Check that G_2 contains all vertices of G_1 ($\alpha_1(x) = \alpha_2(x)\ \forall\ x \in V_1$)

2. For each pair of common vertices check that they are connected with the identical labeled edges ($\beta_1(x, y) = \beta_2(x, y)\ \forall\ (x, y) \in V_1 \times V_1$))

Complexity of the first step is $O(|V1| \times |V2|)$ since we need only to compare each node label of one graph to each node label of another one and determine matching. Complexity of the second step is

$$\binom{|V_1|}{2} + |s| \times |V_1|^2 = \frac{|V_1|!}{(|V_1|-2)!\times 2!} + |s| \times |V_1|^2 = O|V_1|^2$$

where:

$\binom{|V_1|}{2}$ - possible number of pairs

$|s|$ - constant number of sections (three in our standard graph representation) so

$|s| \times |V_1|^2$ is maximal number of edges connecting nodes in $|V_1|$

Total complexity of subgraph isomorphism identification is

$O(|V_1| \times |V_2| + |V_1|^2) \le O(|V_2|^2) = O(|V|^2)$ where $|V| = max(|V_1|, |V_2|)$.

Complexity of graph isomorphism identification is calculated exactly the same way while number of nodes in both graphs is equal ($|V_1| = |V_2|$) so $|V| = |V_1| = |V_2|$.

5 Comparative Evaluation

5.1 Description of Benchmark Data Sets

In order to evaluate the performance of the methods studied in this work, several experiments were performed using four different collections of web documents, called the F-series [2], the J-series [3], the K-series [4] and the U-series [10]. These four document collections were selected for two major reasons. First, all of the original HTML documents are available for these data sets, which is necessary if the web documents are to be represented using the proposed hybrid methodology. Many other text categorization collections provide only a pre-processed vector representation or the plain text, which are both unsuitable for use with our methods. Second, the ground truth class assignments are provided for each data set, with the classes representing easily understandable groupings that relate to the content of the documents. Most web document collections are not labeled or prepared with some other task in mind than content-related classification (e.g., building a predictive model based on user preferences). In this paper, we compare the bag-of-words representation of these four collections with our hybrid techniques using two model based classifiers: Naïve Bayes and C4.5.

The F-series originally contained 98 documents belonging to one or more of 17 subcategories of four major category areas: *manufacturing, labor, business & finance* and *electronic communication & networking*. Since there are multiple subcategory classifications from the same category area for many of these documents, the number of categories was reduced to just the four major categories mentioned above in order to simplify the problem. There were five documents that had conflicting classifications (i.e., they were classified to belong to two or more of the four major categories) which were removed, leaving a total of 93 documents.

The J-series contains 185 documents and ten classes: *affirmative action, business capital, information systems, electronic commerce, intellectual property, employee rights, materials processing, personnel management, manufacturing systems,* and *industrial partnership*. This data set has not been modified for this study.

The K-series consists of 2,340 documents and 20 categories: *business, health, politics, sports, technology, entertainment, art, cable, culture, film, industry, media, multimedia, music, online, people, review, stage, television,* and *variety*. The last 14 categories are subcategories related to entertainment, while the entertainment category refers to entertainment in general. Experiments on this data set are presented in [34]. These were originally news pages hosted at Yahoo (www.yahoo.com). The F, J and K series data sets can be downloaded from the following FTP directory: ftp://ftp.cs.umn.edu/dept/users/boley/PDDPdata.

The U-series is the largest dataset used in this study. It contains 4,167 documents taken from the computer science department of four different universities: Cornell, Texas, Washington, and Wisconsin. Previously documents were divided into seven different categories: *course, faculty, students, project, staff, department* and *other* that catch all other documents. For the classification experiments only four of these classes were used: *course, faculty, students,* and *project*, and the remaining examples were pooled into a single *other* class. This collection can be found and downloaded at http://www.cs.cmu.edu/~webkb.

5.2 Preprocessing and Representation

The following preprocessing steps were applied before using the bag-of-words and hybrid representation techniques:

- All meaningless words ("stopwords") were removed from each document using the list of stopwords given in Appendix B of 33].
- Stemming was done using the Porter stemmer [27].

To construct a dictionary for the bag-of-words representation, N most frequent words were selected from each document. Unique words were inserted in the dictionary. The TF (term frequency) approach to word selection is more efficient than the TF-IDF (term frequency – inverted document frequency) scheme, which is popular in the information retrieval [31], since it does not require recalculation of all term weights with an addition of every new document to the training corpus. The different values of N which were used in these experiments together with the dictionary sizes obtained for the bag-of-words representation can be found in Table 2. Each document was then represented as a vector of Boolean values, 1 for presence and 0 for absence of a dictionary word in the document. The simplest, Boolean representation was chosen to save the preprocessing computations and produce more compact classification models that should maximize the classification speed of unlabeled documents. As can be understood from Table 2, the longest vector (21,463 words) was obtained using the U-series data set and $N = 100$. After the representation stage, a classification model was induced from the training documents and applied to the documents in a validation set. Since in some cases the accuracy gap between two different cross validation runs with the same dictionary can reach 1.5 – 2%, the average of ten runs of a ten-fold cross validation was used as the final accuracy result

for each collection and dictionary size.

Table 2: ***Dictionary sizes per data set and N***

N / *Series*	20	30	40	50	100
F	846	1135	1422	1695	2774
J	1301	1773	2173	2518	3956
K	6553	7988	9258	10663	16874
U	9313	11814	13911	15594	21463

As for Hybrid techniques, the same N most frequent words in each document were taken for graph construction, that is exactly the same words in the document were used for both the graph creation and the bag-of-words representation. Subgraphs relevant for classification were then extracted using the Naïve, the Smart, and the Smart with Fixed Threshold approaches. A dictionary containing subgraphs instead of simple words was constructed. Each document was then represented as a vector of Boolean values, 1 for presence and 0 for absence of a dictionary term (subgraph) in the document. Hundreds of experiments were performed, with N being varied together with t_{min} and CR_{min} for the Naïve and the Smart approaches, respectively.

The Smart with Fixed Threshold approach was applied by defining the fixed threshold $t_{min} = 0.1$ together with CR_{min}, and performing smart extraction. The value of t_{min} was not chosen arbitrarily. The assumption was that subgraphs that appear in less than 10% of the graphs cannot be attributes.

5.3 Comparison of Hybrid and Bag-of-words Representations Using the C4.5 Classifier

Only the best results for each technique are presented here. Classification results for F, J, K and U-series are given in Figures 3, 4, 5, and 6, respectively.

As can be seen from the figures, in all cases the Hybrid approaches achieved better classification results than the regular Vector Space Model (bag-of-words) representation, especially in F, J and U-series data sets where all Hybrid representations showed much better results for all values of N. In the K-series data set case (Figure 5), the bag-of-words representation outperformed the Hybrid Smart and the combined approach for some values of N. However, the best classification accuracy was still found to belong to the Hybrid Naïve method. The values of input parameters together with the subgraph dictionary sizes for the best accuracy results are shown in Table 3. The best accuracy results for each data set across all methods are emphasized in bold and it can be easily seen that they were all produced by the hybrid techniques with mostly minor differences between various hybrid representations. Using the Normal approximation to the Binomial distribution, all best accuracy results of the hybrid methods were found

significantly higher than the best results of the bag-of-words representation (at the significance level lower than 0.001).

[Figure 3: Comparative results for F-series with C4.5 classifier]

[Figure 4: Comparative results for J-series with C4.5 classifier]

[Figure 5: Comparative results for K-series with C4.5 classifier]

[Figure 6: Comparative results for U-series with C4.5 classifier]

Table 3: ***Input parameters and obtained dictionary sizes for the best accuracy results*** **(C4.5)**

Data Set	*Method*	*Number of Frequent Words Used for Dictionary or Graph Creation* ***N***	*Minimal Subgraph Frequency Threshold tmin*	*Minimal Classification Rate CRmin*	*Dictionary Size*	*Accuracy*
F-series	Hybrid Smart	50	n/a	0.8	411	86.56%
	Hybrid Naïve	50	0.2	n/a	152	86.02%
	Hybrid with Fixed Threshold	**50**	**0.1**	**0.8**	**300**	**88.6%**
	Bag-of-words	20	n/a	n/a	846	78.06%
J-series	**Hybrid Smart**	**30**	**n/a**	**1.2**	**2503**	**85.24%**
	Hybrid Naïve	20	0.15	n/a	1668	84.65%
	Hybrid with Fixed Threshold	20	0.1	0.7	1635	83.41%
	Bag-of-words	50	n/a	n/a	2518	58.32%
K-series	Hybrid Smart	20	n/a	1.1	2102	74.68%
	Hybrid Naïve	**100**	**0.25**	**n/a**	**24.35**	**78.18%**
	Hybrid with Fixed Threshold	100	0.1	1.4	3644	73.86%
	Bag-of-words	50	n/a	n/a	10663	73.01%
U-series	**Hybrid Smart**	**100**	**n/a**	**1.1**	**64**	**82.44%**
	Hybrid Naïve	100	0.1	n/a	360	81.75%
	Hybrid with Fixed Threshold	100	0.1	1	80	82%
	Bag-of-words	20	n/a	n/a	9313	78.11%

The times needed to build a classification model and categorize one document in the U-series data set, which was the most time-consuming task in our experiments, were also

measured and compared. The time required for each procedure was measured on the same system under the same operation conditions: a 2GHz Pentium 4 processor with one Gigabyte of RAM. Execution time was measured for the most accurate cases (see Table 3) of each approach. The total time span needed to create a classification model with each method is given in Table 4. It is called the offline classification time because the classification model is usually constructed before the massive categorization stage and does not change in the process[5]. The model generation stage consists of the following steps applied to the training collection of documents:

1. *Time to Build Graphs* – time needed to build graphs from each document in the collection (not relevant for the bag-of-words representation)
2. *Time to Build Dictionary* – for the Hybrid technique this is the time needed to extract relevant subgraphs, while for the bag-of-words it is the time needed to find and combine the most frequent words from every document.
3. *Time to Construct Vectors* – time required for documents representation in the vector format.
4. *Time to Induce Classification Model* – inducing a classification model from the feature vectors with a classification algorithm (e.g., C4.5 or Naïve Bayes Classifier).

It is interesting to note that the extraction process using the Smart method took much more time than the Naive Hybrid technique. Such a difference occurred because an infrequent subgraph cannot be dropped without calculating its *CR*. Another fact which catches one's attention is that creating a dictionary using the hybrid approach with fixed threshold (subgraphs extraction) is faster than creating a dictionary for bag-of-words, even for relatively small *N* (20 in this case). All Hybrid techniques also demonstrated faster document representation and model creation times than the bag-of-words representation. This fact can be easily explained by the size of the dictionary obtained using the hybrid approaches, which is much smaller than the dictionary used with the bag-of-words representation (see Table 3). Finally, the shortest total time is reached with the Hybrid Smart approach using a fixed threshold, where nearly the highest accuracy is also reached.

The average time needed to classify one document, or *online classification time,* for current cases is presented in Table 5. This parameter is much more important than the model generation time when real-time categorization of massive web document streams is required. As can be seen, documents represented by the hybrid techniques are classified much faster than documents represented as “bags of words”. This is due to the relatively small dictionary size and the resultant smaller decision-tree model.

[5] *Incremental* induction of classification models is beyond the scope of this research

Table 4: ***Total time needed to induce a classification model (C4.5)***

Data Set	Method	Time to Build Graphs (sec)	Time to Build Dictionary (sec)	Time to Construct Vectors (sec)	Time to Build Classification Model (sec)	Total Time Offline (sec)
U-series	Hybrid Smart	223.2	2628.56	5.59	4.36	2861.71
	Hybrid Naïve	223.2	43.4	31.16	76.59	374.35
	Hybrid with Fixed Threshold	223.2	66.35	7.47	6.09	303.11
	Bag-of-words	n/a	300.9	133.2	330.32	764.42

Table 5: Average time to classify one document (C4.5) – U Series

Method	Average Time to Classify One Document(sec)
Hybrid Smart	2.88×10^{-4}
Hybrid Naïve	4.56×10^{-4}
Hybrid with Fixed Threshold	3.12×10^{-4}
Bag-of-words	1.68×10^{-3}

In this study, we were also interested to explore in more details the added value of multi-node subgraphs, since single-node subgraphs are equivalent to the bag-of-words model. We define a multi-node graph as the one that contains two or more nodes. Obviously, the presence of such graphs into a term set *T* makes the difference between the hybrid and bag-of-words representations. In Figures 7-10 we show the percentage of multi-node subgraphs in term sets *T* for the best classification accuracy results in each document collection. As can be seen in Figure 7, for instance, with the Hybrid Naïve representation and 20 node document graph size we have more than 50% multi-node subgraphs in the term set *T*. It is noteworthy that a relatively high percentage of multi-node subgraphs was found in J and K series collections so the impact of multi-node subgraphs in those collections was high too. As a rule we can say that the amount and the resulting impact of multi-node subgraphs were found significant in most cases.

[Figure 7: Percentage of Multi-Node Sub-Graphs in T for F-Series (Best C4.5 results)

[Figure 8: Percentage of Multi-Node Graphs in T for C4.5, J-Series (Best C4.5 results)]

[Figure 9: Percentage of Multi-Node Graphs in T for C4.5, K-Series (Best C4.5 results)]

[Figure 10: Percentage of Multi-Node Graphs in T for C4.5, U-Series (Best C4.5 results)]

5.4 Comparison of Hybrid and Bag-of-words Representations Using Probabilistic Naïve Bayes Classifier

In the experiments using the Naïve Bayes Classifier (NBC) the same preprocessing stages as in the previous section were performed. Exactly the same input parameter values and document representations were used in the empirical evaluation. Since the document representation stage remained unchanged, the dictionaries used for the C4.5 runs stayed exactly the same. Accuracy results for the F, J, K and U-series collections are presented in Figures 11-14, respectively. An overview of the best accuracy results along with the corresponding input parameters and dictionary sizes are given in Table 6. The best accuracy results for each data set across all methods are again emphasized in bold. In case of the Naïve Bayes (NBC), the hybrid techniques are better than the bag-of-words representation for three document collections out of four (F, J, and U-series). All differences are statistically significant at the level lower than 0.001. However, for the K-series collection (see Figure 13), the bag-of-words representation achieved slightly better accuracy results for most values of *N*. This may be explained by the NBC's ability to perform well with a large number of uncorrelated or weakly correlated features.

Table 6: ***Input parameters and obtained dictionary sizes for the best accuracy results (NBC)***

Data Set	*Method*	*Number of Frequent Words Used for Dictionary or Graph Creation* ***N***	*Minimal Subgraph Frequency Threshold tmin*	*Minimal Classification Rate CRmin*	*Dictionary Size*	*Accuracy*
F-series	Hybrid Smart	50	1.2	n/a	108	94.84%
	Hybrid Naïve	50	0.2	n/a	152	94.84%
	Hybrid with Fixed Threshold	**50**	**0.1**	**1.3**	**93**	**95.27%**
	Bag-of-words	20	n/a	n/a	846	91.16%
J-series	Hybrid Smart	100	n/a	1.8	347	84.49%
	Hybrid Naïve	**100**	**0.4**	**n/a**	**202**	**90.7%**
	Hybrid with Fixed Threshold	30	0.1	0.9	567	83.46%
	Bag-of-words	20	n/a	n/a	1301	57.68%
K-series	Hybrid Smart	100	n/a	1.8	891	75.18%
	Hybrid Naïve	20	0.15	n/a	497	73.55%
	Hybrid with Fixed Threshold	100	0.1	0.9	1575	75.43%
	Bag-of-words	**20**	**n/a**	**n/a**	**6553**	**76.97%**
U-series	**Hybrid Smart**	**100**	**n/a**	**1.2**	**48**	**78.97%**
	Hybrid Naïve	20	0.2	n/a	24	76.53%
	Hybrid with Fixed Threshold	100	0.1	1.2	48	78.97%
	Bag-of-words	100	n/a	n/a	21463	72.59%

[Figure 11: Comparative results for F-series with Naïve Bayes Classifier]

[Figure 12: Comparative results for J-series with Naïve Bayes Classifier]

[Figure 13: Comparative results for K-series with Naïve Bayes Classifier]

[Figure 14: Comparative results for U-series with Naïve Bayes Classifier]

Comparative timing results for classification model creation and single document categorization are given in Tables 6 and 7, respectively. The classification model induction took more time with the Hybrid Smart extraction than with the bag-of-words, but other hybrid techniques succeeded in performing faster. Computational time reduction with other hybrid approaches is explained by the presence of a fixed frequency threshold that helps to remit the number of candidate subgraphs. A significant improvement is also seen in the average time required to classify one document using all hybrid approaches vs. the bag-of-words, which, of course, results from a smaller dictionary size, obtained by the hybrid methods. The percentages of multi-node subgraphs in term sets *T* that provided the best NBC accuracy results in each document collection were similar to the best term sets used by the C4.5 algorithm and we do not show them here due to space limitations.

Table 7: ***Total time needed to create classification model (NBC)***

Data Set	Method	Time to Build Graphs(sec)	Time to Build Dictionary(sec)	Time to Construct Vectors(sec)	Time to Build Classification Model(sec)	Total Time Offline(sec)
U-series	Hybrid Smart	223.2	2460.87	4.21	0.12	2688.4
	Hybrid Naïve	283.64	1.46	0.5	0.08	285.68
	Hybrid with Fixed Threshold	223.2	62.3	4.19	0.12	289.81
	Bag-of-words	n/a	51.55	286.34	42.62	380.51

Table 8: ***Average time to classify one document (NBC) – U Series***

Method	Average Time to Classify One Document(sec)
Hybrid Smart	1.2×10^{-3}
Hybrid Naïve	6.49×10^{-4}
Hybrid with Fixed Threshold	5.7×10^{-4}
Bag-of-words	0.125

6 Conclusions and Future Research

The goal of this research study was to propose novel graph-based document representations for efficient categorization of web documents with model-based classifiers. An important objective was also to maintain or even improve the prediction performance of document categorization. The proposed hybrid representation models were shown to be considerably faster than the traditional vector-space model in terms of online classification speed, while also outperforming the predictive accuracy of the same model, in most cases. Thus, the techniques described here can now be used for real-time document categorization applications.

As for future research, some issues are still open:

1. Some heuristic should be developed for finding the optimal representation model and its input parameters (like N, CR_{min} and t_{min}) for a given classification task. The evaluation in this research study was purely empirical, and the experiments were run using a wide range of input values for three different hybrid representations to achieve better classification results. Despite the relatively small accuracy difference between the best and the worst classification results, there is still a need in some simple heuristic.
2. In this research, the ability of the hybrid representations to perform a document classification task was examined. However, we hope that the techniques presented in this work can also be easily applied to other web content mining tasks such as web document clustering. This capability should be further explored and evaluated.
3. The techniques presented here can be tested with other classifiers, beyond C4.5 and NBC that may provide even better results. They can also be used together with other representation techniques. A good example would be the relational representation [9, 10], where relevant subgraphs can be used as background knowledge instead of simple words.

Acknowledgment

This work was partially supported by the National Institute for Systems Test and Productivity at the University of South Florida under the USA Space and Naval Warfare Systems Command Grant No. N00039-01-1-2248.

References

1. C. Apt'e, F. J. Damerau and S. M. Weiss, "Automated learning of decision rules for text categorization", ACM Transactions on Information Systems 12, 3, 233–251, 1994.
2. D. L. Boley, "Principal Direction Divisive Partitioning", Data Mining and Knowledge Discovery 2(4):325-344, 1998.
3. D. Boley, M. Gini, R. Gross, S. Han, K. Hastings, G. Karypis, V. Kumar, B. Mobasher, J. Moore, "Partitioning-Based Clustering for Web Document Categorization", Decision Support Systems, 1999.
4. D. Boley, M. Gini, R. Gross, E.-H. (Sam) Han, K. Hastings, G. Karypis, V. Kumar, B. Mobasher, J. Moore, "Document Categorization and Query Generation on the World Wide Web Using WebACE", AI Review, 13:365-391, 1999.

5. H. Bunke, "On a relation between graph edit distance and maximum common subgraph", Pattern Recognition Letters, Vol.18, 1997, pp.689–694.
6. H. Bunke, "Error Correcting Graph Matching: On the Influence of the Underlying Cost Function", IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol.21, No.9, September 1999, pp.917–922.
7. R. Carreira, J. M. Crato, D. Goncalves, and J. A. Jorge, "Evaluating Adaptive User Profiles for News Classification", Proc. 9th International Conference on Intelligent User Interface, January 2004.
8. S. Chakrabarti, B. Dom, and P. Indyk, "Enhanced hypertext categorization using hyperlinks", in *Proceedings of the 1998 ACM SIGMOD international Conference on Management of Data* (Seattle, Washington, United States, June 01 - 04, 1998). A. Tiwary and M. Franklin, Eds. SIGMOD '98. ACM Press, New York, NY, pp. 307-318, 1998.
9. W.W. Cohen, "Learning to classify English text with ILP methods", In Advances in Inductive Logic Programming (Ed. L. De Raedt), IOS Press, 1995.
10. M. Craven, D. DiPasquo, D. Freitag, A. McCallum, T. Mitchell, K. Nigam and S. Slattery, "Learning to extract symbolic knowledge from the World Wide Web", In Proceedings of the Fifteenth National Conference on Artificial Intellligence (AAAI98), pages 509-516, 1998.
11. J. Fagan, "Experiments in Automatic Phrase Indexing for Document Retrieval: A Comparison of Syntactic and Non-Syntactic Methods", PhD thesis, Dept. of Computer Science, Cornell University, 1987.
12. P. Flach, "The logic of learning: a brief introduction to Inductive Logic Programming", Technical Report: CS-EXT-1998-141, 1988.
13. S. Fortin, "The graph isomorphism problem", Technical Report TR96-20, Department of Computing Science, University of Alberta, 1996.
14. M. R. Garey, D. S. Johnson, "Computers and Intractability: A Guide to the Theory of NP-Completeness", W. H. Freeman and Company, New York, 1979.
15. H.George, J&P. Langley, "Estimating Continuous Distributions in Bayesian Classifiers", Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence. pp. 338-345. Morgan Kaufmann, San Mateo.
16. H. Hall, "Networked information: dealing with overload", Proceedings of Information 1997 (Strathclyde Business School, November 1997), Library Association CIG Scotland, Paisley, 37-44.
17. A. Hotho, S. Staab, and G. Stumme, "Wordnet improves Text Document Clustering", In *Proc. of the Semantic Web Workshop of the 26th Annual International ACM SIGIR Conference*, Toronto, Canada, 2003.
18. D. Koller, and M. Sahami, "Hierarchically classifying documents using very few words", In Proceedings of ICML-97, 14th International Conference on Machine Learning, (Nashville, US, 1997), pp. 170–178, 1997.
19. M. Kuramochi and G. Karypis, An Efficient Algorithm for Discovering Frequent Subgraphs. IEEE Transactions on Knowledge and Data Engineering 16, 9 (Sep. 2004), pp. 1038-1051.
20. D. D. Lewis, 1992a, "An evaluation of phrasal and clustered representations on a text categorization task", In Proceedings of SIGIR-92, 15th ACM International Conference on Research and Development in Information Retrieval, pp. 37–50, 1992.

21. N. Maria, and M. J. Silva, "Theme-based Retrieval of Web news", Proc. 23rd Annual International ACM SIGIR Conference on Research and Development In Information Retrieval, July 2000.
22. A. Markov and M. Last, "A Simple, Structure-Sensitive Approach for Web Document Classification", in P.S. Szczepaniak et al. (Eds.), Advances in Web Intelligence, Proceedings of the 3rd Atlantic Web Intelligence Conference (AWIC 2005), Springer-Verlag, LNAI 3528, pp. 293–298, Berlin Heidelberg 2005.
23. A. Markov and M. Last, "Efficient Graph-Based Representation of Web Documents", Proceedings of the Third International Workshop on Mining Graphs, Trees and Sequences (MGTS2005), pp. 52-62, October 7, 2005, Porto, Portugal.
24. A. Markov, M. Last, and A. Kandel, "Model-Based Classification of Web Documents Represented by Graphs", Proceedings of the WebKDD 2006 Workshop on Knowledge Discovery on the Web at KDD 2006, pp. 31-38, Philadelphia, PA, USA, Aug. 20, 2006.
25. A. McCallum, and K. Nigam, "A Comparison of Event Models for Naive Bayes Text Classification", Proc. AAAI--98 Workshop on Learning for Text Categorization, 1998.
26. F. Peng, D. Schuurmans, "Combining Naive Bayes and n-Gram Language Models for Text Classification", in Advances in Information Retrieval: Proc. 25th European Conference on IR Research, ECIR 2003, Pisa, Italy, April 14-16, pp. 335 – 350, 2003.
27. M. Porter, "An algorithm for suffix stripping", Program Vol. 14, No. 3, 130-137, 1980.
28. J. R. Quinlan, R. M. Cameron-Jones, "Induction of logic programs: FOIL and related systems", New Generation Computing, 13(3, 4), 287–312, 1995.
29. J.R. Quinlan, "C4.5: Programs for Machine Learning", Morgan Kaufmann Publishers Inc, 1993.
30. G. Salton, A. Wong, and C. Yang, C. (1975). A Vector Space Model for Automatic Indexing, J. Communications of the ACM, 18(11), pp. 613--620.
31. G. Salton, and C. Buckley, "Term Weighting Approaches in Automatic Text Retrieval", Technical Report: TR87-881, Cornell University, November 1988.
32. G. Salton, and M. McGill, "Introduction to Modern Information Retrieval", McGraw Hill, 1983.
33. A. Schenker, H. Bunke, M. Last, and A. Kandel, "Graph-Theoretic Techniques for Web Content Mining", Series in Machine Perception and Artificial Intelligence, 62, World Scientific, 2005.
34. A. Strehl, J. Ghosh, and R. J. Mooney, "Impact of similarity measures on web-page clustering", In Proc. AAAI Workshop on AI for Web Search (AAAI 2000), Austin, pages 58-64. AAAI/MIT Press, July 2000.
35. C.-M. Tan, Y.-F. Wang, and C.-D. Lee, "The use of bigrams to enhance text categorization", Information Processing and Management, Vol. 38, 2002, pp. 529–546.
36. S. M. Weiss, C. Apte, F. J. Damerau, D. E. Johnson, F. J. Oles, T. Goetz and T. Hampp, "Maximizing Text-Mining Performance", J. IEEE Intelligent Systems, 14(4), July/August, pp. 63—69, 1999.
37. X. Yan and J. H. Gspan, "gSpan: Graph-Based Substructure Pattern Mining", In *Proceedings of the 2002 IEEE international Conference on Data Mining (ICDM'02)*

(December 09 - 12, 2002). ICDM. IEEE Computer Society, Washington, DC, pp. 721-724.

Figures

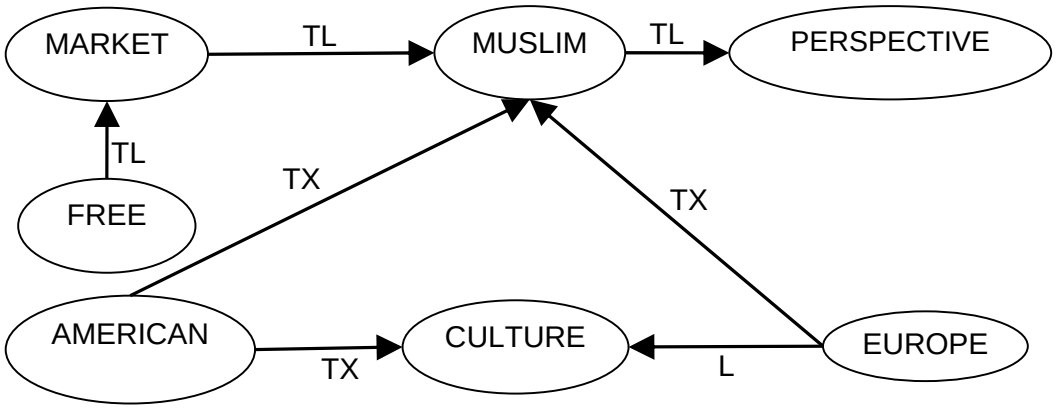

Figure 1: Standard Graph Document Representation

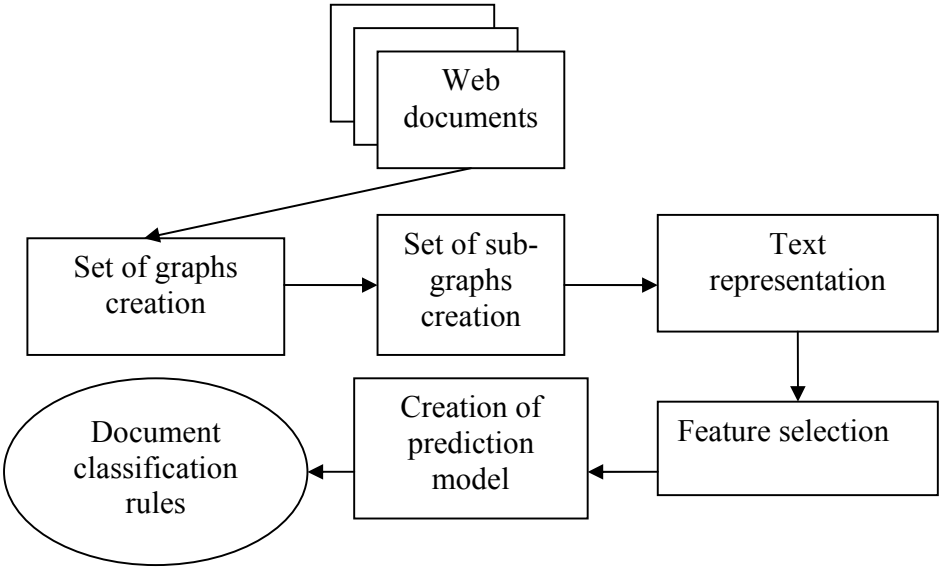

Figure 2: Classification Model Induction

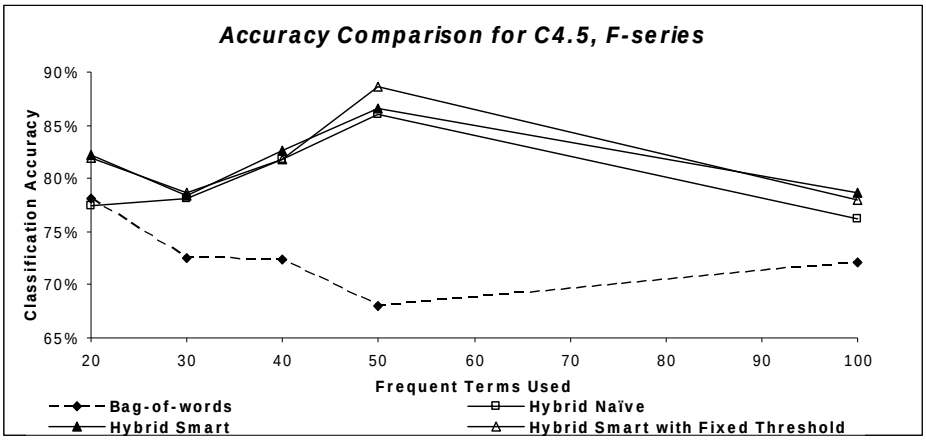

Figure 3: ***Comparative results for F-series with C4.5 classifier***

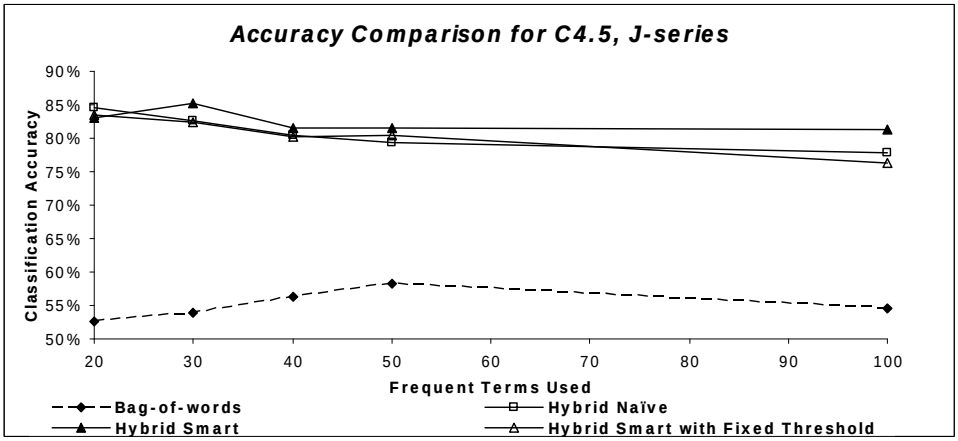

Figure 4: ***Comparative results for J-series with C4.5 classifier***

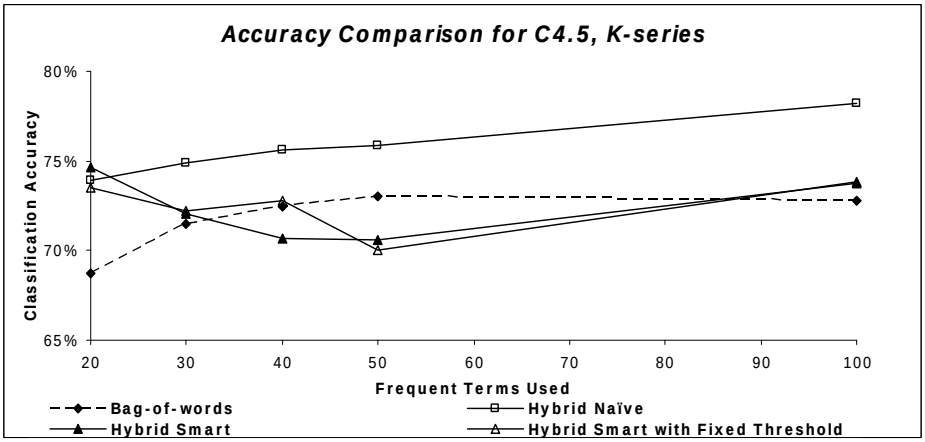

Figure 5: ***Comparative results for K-series with C4.5 classifier***

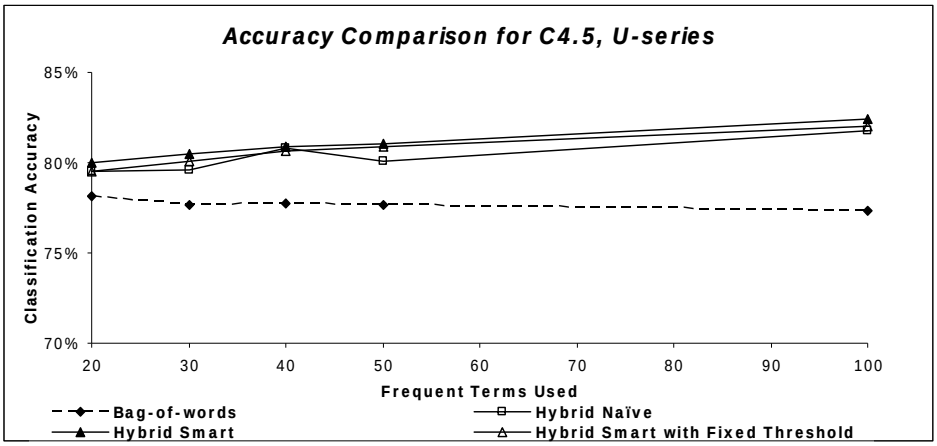

Figure 6: ***Comparative results for U-series with C4.5 classifier***

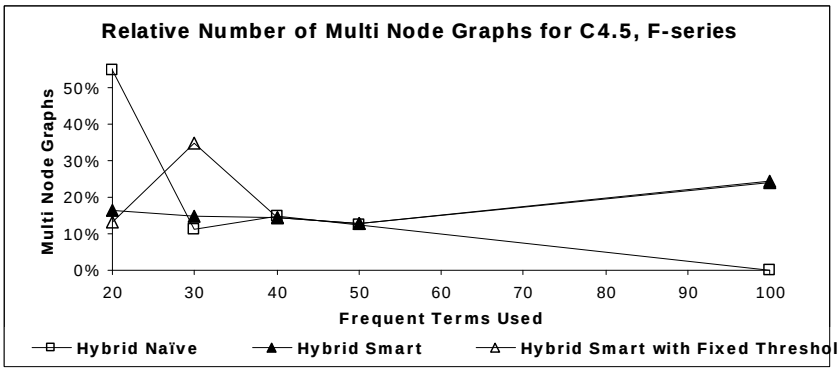

Figure 7: ***Percentage of Multi-Node Sub-Graphs in T for F-Series (Best C4.5 results)***

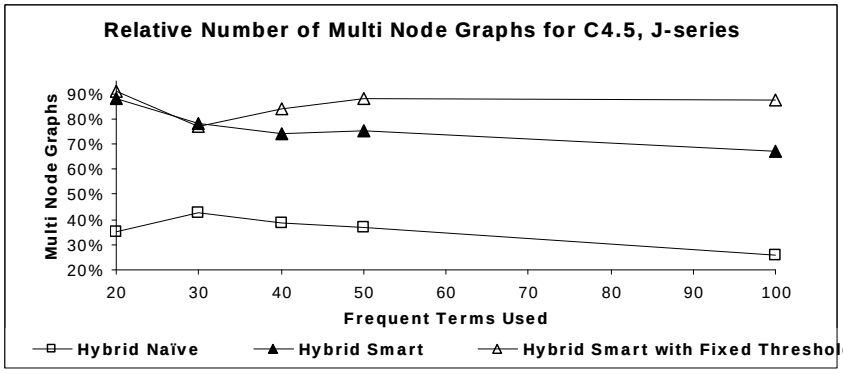

Figure 8: ***Percentage of Multi-Node Graphs in T for C4.5, J-Series (Best C4.5 results)***

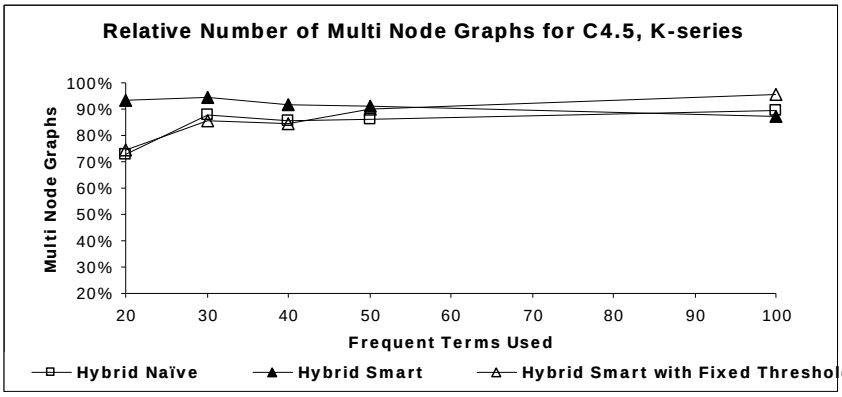

Figure 9: ***Percentage of Multi-Node Graphs in T for C4.5, K-Series (Best C4.5 results)***

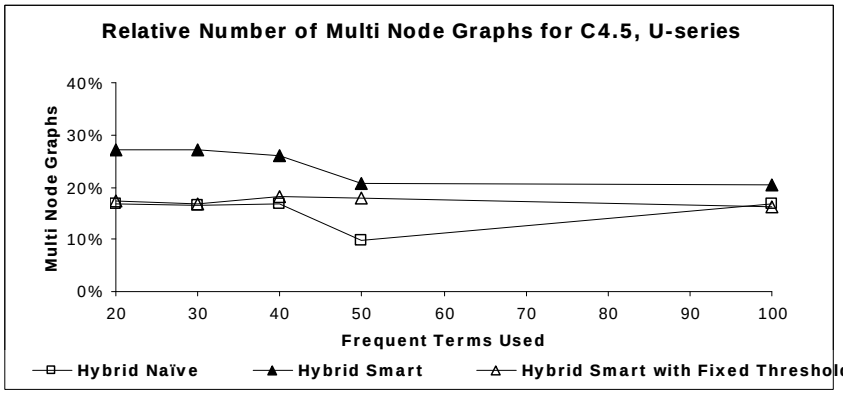

Figure 10: ***Percentage of Multi-Node Graphs in T for C4.5, U-Series (Best C4.5 results)***

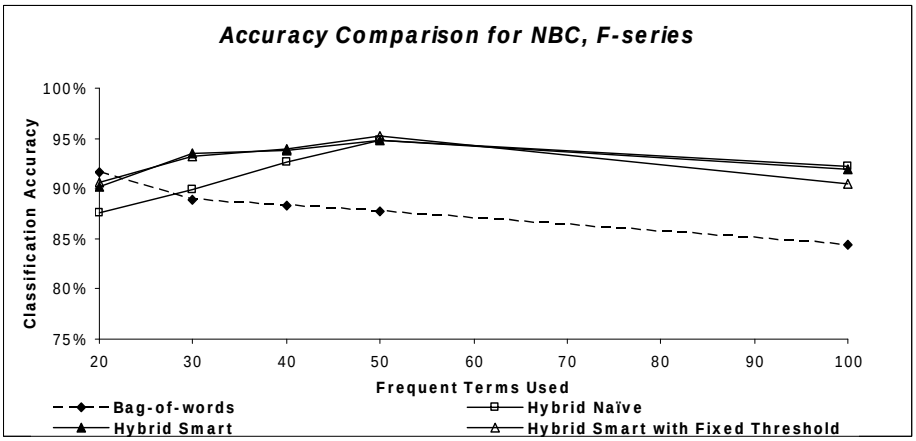

Figure 11: ***Comparative results for F-series with Naïve Bayes Classifier***

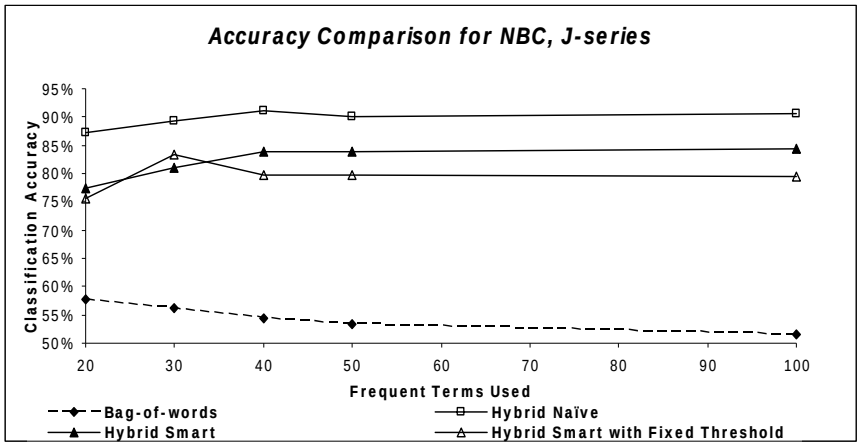

Figure 12: ***Comparative results for J-series with Naïve Bayes Classifier***

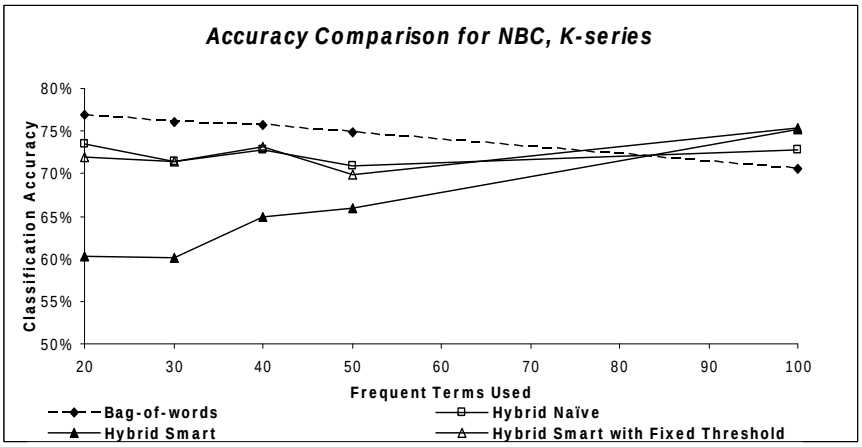

Figure 13: ***Comparative results for K-series with Naïve Bayes Classifier***

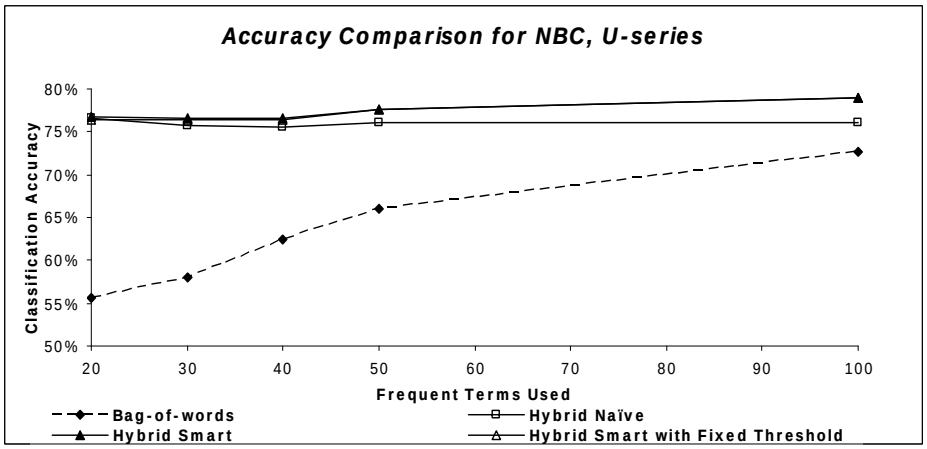

Figure 14: ***Comparative results for U-series with Naïve Bayes Classifier***