Term Graph Model for Text Classification*

Wei Wang, Diep Bich Do, and Xuemin Lin

University of New South Wales, Australia
{weiw, s2221417, lxue}@cse.unsw.edu.au

Abstract. Most existing text classification methods (and text mining methods at large) are based on representing the documents using the traditional vector space model. We argue that important information, such as the relationship among words, is lost. We propose a term graph model to represent not only the content of a document but also the relationship among the keywords. We demonstrate that the new model enables us to define new similarity functions, such as considering rank correlation based on PageRank-style algorithms, for the classification purpose. Our preliminary results show promising results of our new model.

1 Introduction

In recent years, text mining has become one of the most popular research areas in data mining, due to the rapid growth and evolution of digital text documents, such as Web pages, office documents, and E-mails. As the demand to organize these documents automatically is constantly rising, *text classification* (or *text categorization*) become one active subfields for data mining researchers. Text classification deals with the problem of automatically assigning single or multiple category (or class) labels to a new text document based after learning from a set of training documents with correct category labels.

Most existing text classification methods (and text mining methods at large) adopt the approach of transforming the text mining problem into traditional machine learning problem, where a large number of mature techniques can be applied [7, 14, 17, 18, 8]. Usually, the conversion of a text document into a relational tuple is performed using the popular *vector-space model* model. Intuitively, the document is parsed, cleaned and stemmed, in order to obtain a list of *terms* with corresponding frequencies. Then a corresponding vector can be constructed to represent the document. Therefore, a collection of documents can be represented by a *term-by-frequency* matrix, which can be subsequently interpreted as a relational table.

However, the vector space model only allow preserving fundamental features of the document. Although a few alternative weighting scheme other than term frequency have been proposed, one common weakness is that they don't take into consideration the associations among terms. Recent studies have revealed that association among terms could provide rich semantics of the documents and

* This work was partially supported by ARC Discovery Grant – DP0346004.

X. Li, S. Wang, and Z.Y. Dong (Eds.): ADMA 2005, LNAI 3584, pp. 19–30, 2005.

serve as the basis of a number of text mining tasks [9]. However, the approach proposed in [9] discards the vector space model and uses frequently co-occurring terms only.

In this paper, we propose a novel model for text document by combining the strengths of vector space model and frequently co-occurring terms together. The result is called the *term graph model*. The basic idea is to mine the associations among terms, and then capture all these information in a graph. We use text classification to illustrate the potential application of the new model. To that end, we design two novel similarity functions. One is based on Google's page-rank style algorithm [3] to discover the weights of each terms. The other is based on the distances of all pairs of terms. Our preliminary experimental results shows our new model is promising.

The rest of the paper is organized as follows. Section 2 introduces related works in text classification. Section 3 presents the proposed term graph model which is capable of capturing more information than the vector space model for text documents. In Section 4, we propose methods to classify text documents represented in the term graph model. Experimental results based on the Reuters-21578 text collection is described in section 5. We conclude the paper in Section 6.

2 Related Work

An immense amount of work has been done in the area of text classification in the past decade. We refer readers to [15] for a recent survey. In the rest of this section, we will only focus on several work that is most related to our approach in this section.

The *support vector machine* (SVM) technique — a popular and highly accurate machine learning method for classification problems — was first in introduced in the early 1990s [5]. In 1998, the study proposed by Joachims explored the benefits of using SVM for text categorization [8]. SVM-based approaches can handle large feature spaces with excellent classification accuracy. As a result, SVM-based system has ability to work well for the standard text corpus. The results in [8] shows that SVM-based method is more accurate than alternative approaches. SVM has also been suggested in [14, 18] as one of the most outperforming classifiers in comparison with a set of alternative text categorization methods. One weakness of SVM-based text categorization system is that it cannot scale well with the number of documents in the text collections.

Text categorization based on association rule mining is also another promising approach. [1] proposed two different methods for generating text classifier based on associating the words of a document and its pre-defined categories. These methods are called the Association Rule-based Categorizer By Category (ARC-BC) and the Association Rule-based Categorizer for All Categories (ARC-AC). The main ideas in both approaches are:

1. Present each training document as a transactions of terms.
2. Use a special association rule mining algorithm that is guided by constraints to produce the expected rules in the form of $T \Rightarrow c_i$ where T is set of terms. The results then will be used directly for classification.

The difference between the two method lies in the granularity when forming the text collection. In the ARC-AC algorithm, all the categories form a text collection and the only set of rules generated form the classifier. Meanwhile, ARC-BC algorithm considers each category as a separate text collection and a distinct set of association rules is generated for each category.

More recently, document level frequent itemsets is explored more for other problems like text clustering and learning from the web. Liu et al. [9] introduced a novel system to mine *topic-specific* knowledge on the Web. The intuition is that the frequent word phrases in a collection of web pages of the same topic are most likely to be the sub-topics or the salient concepts. We can find out several different methods and system for clustering transactions [16] and documents [2, 6]. These are all based on the intuition that there should be many frequent itemsets within a cluster and different clusters have little overlapping of such frequent itemsets.

3 Term Graph Model

3.1 Overview of the Term Graph Model

The term graph model is an improved version of the vector space model [13] by weighting each term according to its relative "importance" with regard to term associations. Specifically, for a text document D_i, it is represented as a vector of term weights $\boldsymbol{D_i} =< w_{1i}, \ldots, w_{|T|i} >$, where T is the ordered set of terms that occur at least once in at least one document in the collection. Each weight w_{ji} represents how much the corresponding term t_j contribute to the semantics of document d_i. Although a number of weighting schemes have been proposed (e.g., boolean weighting, frequency weighting, tf-idf weighting, etc.), those schemes determine the weight of each term *individually*. As a result, important yet rich information regarding the relationships *among* the terms are not captured in those weighting schemes.

We propose to determine the weight of each term in a document collection by constructing a *term graph*. The basic steps are as follows:

1. *Prepocessing Step*: For a collection of document, extract all the terms.
2. *Graph Building Step*:
   (a) For each document, we view it as a transaction: the document ID is the corresponding transaction ID; the terms contained in the document are the items contained in the corresponding transaction. Association rule mining algorithms can thus be applied to mine the frequently co-occurring terms that occur more than *minsup* times in the collection.
   (b) The frequent co-occurring terms are mapped to a weighted and directed graph, i.e., the *term graph*.

We will introduce the details of each step as follows.

3.2 Preprocessing

In our term graph model, we will capture the relationships among terms using the frequent itemset mining method. To do so, we consider each text document in the

training collections as a transaction in which each word is an item. However, not all words in the document are important enough to be retained in the transaction. To reduce the processing space as well as increase the accuracy of our model, the text documents need to be preprocessed by (1) remove stopwords, i.e., words that appear frequently in the document but have no essential meanings; and (2) retaining only the root form of words by stemming their affixes as well as prefixes. We use Lancaster algorithm for stemming [11].

3.3 Graph Building

As mentioned above, we will capture the relationships among terms using the frequent itemset mining method. While this idea has been explored by previous research [9], our approach distinguish from previous approaches in that we maintain all such important associations in a graph. The graph not only reveals the important semantics of the document, but also provide a basis to extract novel features about the document, as we will shown in the next section.

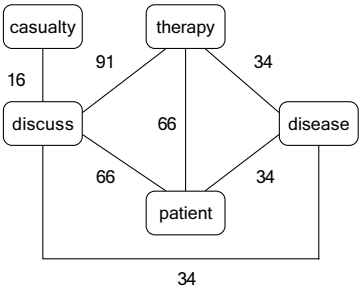
Frequent Itemset Mining. After the preprocessing step, each document in the text collection will be stored as a transaction (list of items) in which each item (term) is represented by a unique non-negative integer. Then frequent itemset mining algorithms can be used to find all the subset of items that appeared more than a threshold amount of times (controlled by *minsup*) in the collection. In our implementation, we use the AFOPT algorithm [10].

Graph Builder. In our system, our goal is to explore the relationships among the important terms of the text in a category and try to define a strategy to make use of these relationships in the classifier and other text mining tasks. Vector space model cannot express such rich relationship among terms. Graph is thus the most suitable data structure in our context, as, in general, each term may be associated with more than one terms.

We propose to use the following simple method to construct the graph from the set of frequent itemsets mined from the text collections. First, we construct a node for each unique term that appear at least once in the frequent itemsets.

Itemset	Support
$\{therapy, discuss\}$	91
$\{therapy, discuss, patient\}$	66
$\{therapy, discuss, patient, disease\}$	34
$\{casualty, discuss\}$	16

(a) Frequent Itemsets



(b) The Corresponding Graph

Fig. 1. An Example Term Graph

Then we create edges between two node u and v if and only if they are both contained in one frequent itemset. Furthermore, we assign weights to the edges in the following way: the weight of the edge between u and v is the largest support value among all the frequent itemsets that contains both of them.

Example 1. Consider the frequent itemsets and their absolute support shown in Figure 1(a). Its corresponding graph is shown in Figure 1(b).

4 Text Classification

Our term graph model encapsulates richer information than the traditional vector space model. As we preserve and extract the hidden relationships among terms in the document collection, we argue that many text mining applications can benefit from this model. Specifically, we consider the classic text classification problem with our new model.

One central notion to the classification is the similarity (or distanct) function of a document and a category. We consider two different approaches based on the term graph model. In the first approach, we borrow the idea of PageRank ranking of the web pages [3] to assign weights to the nodes (i.e., terms) in the term graph; we can then measure the similarity of a document and a category using a rank correlation coefficient [12] based on the ranks of the terms. In the second approach, we define a similarity formula based on the distance matrix of the term graph. More details about those two approaches are described in the following sub-sections.

4.1 Classification Based on the Term Ranks

Ranking Terms. PageRank is a well-known method for measuring the relative importance of the web pages based on their linking information. According to [3], the basic intuition of the PageRank is that a page will have a high rank if there are many pages in the web point to it, or if there are some pages with high ranks pointing to it. By following the same idea, we can determine the "PageRank" scores for the nodes in the term graph (or a document or a category) too. Intuitively, if a word that appears frequently with many other words in the text collections, it is an important word; words that appear together with some important words may also be important.

Since the original Pagerank computation algorithm accept as input a directed, unweighted graph, we need to use the following transformation on our term graph:

- treat each node in the term graph as a web page.
- treat each edge between node u and v with weight w in the graph as $2w$ links; w of them are $u \rightarrow v$ and the other w links as $v \rightarrow u$.

The output can be directly feed into the PageRank computation algorithm. An example of the term graph with PageRank scores computed for each node can be found in Figure 4.

Rank Correlation Coefficient. After calculating the rank of the nodes in the graph using the idea of PageRank, each term a the document is assigned a PageRank value. One simple method to calculate the similarity between a document and a category is as follows: we directly use the PageRank values of the terms as their weights and existing document similarity functions (for instance, the cosine similarity) can be directly applied. In this sense, we can view the process of constructing the term graph and calculating the PageRank for each term as an preprocessing step to obtain yet another *weighting scheme*.

However, we argue that the relative order rather than the absolute values of the PageRank scores are meaningful. Therefore, we propose another similarity metric based on the concept of rank correlation. The basic idea is that if a document belongs to a category, the relative order of terms appearing in the document and the documents belonging to the category should be consistent.

To compute the rank correlation coefficient, we need to obtain the rank of each term. This can be computed by sorting the terms by their PageRank scores in the descending order. We can do this for a (testing) document as well as a category, where we treat all the documents belonging to the category in question as a single document.

It is well-known that a robust statistics to measure the correlation of two arrays of size N is the non-parametric correlation scores, for example, the Spearman Rank-Order Correlation Coefficient [12]. Specifically, let R_i be the rank of x_i among the other x's, S_i be the rank of y_i among the other y's. Then the rank-order correlation coefficient is defined to be the linear correlation coefficient of the ranks, namely,

$$r_s = \frac{\sum_i^N (R_i - \overline{R})(S_i - \overline{S})}{\sqrt{\sum_i^N (R_i - \overline{R})^2}\sqrt{\sum_i^N (S_i - \overline{S})^2}} \tag{1}$$

However, there ordered terms of a document and a category are usually of different length. Therefore, Equation 1 cannot be directly applied. We propose the following heuristics to solve this problem, as shown in Figure .

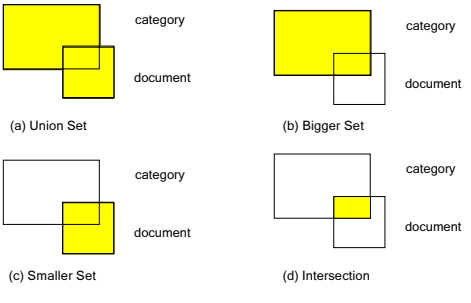


Fig. 2. Four Heuristics to Generate the Rank Correlation Cofficient

1. **Union Set.** In this heuristic, we consider all the terms from the document and the category. In order to calculate the rank correlation, we need to assign rank values to terms that do not appear in the document (or the category). We simply assign the same rank to all the terms that are unique to one input For example, suppose there are $p + n$ terms for the document and $p + m$ terms for the category (that is, there are p terms that appear in both the document and the category). We will find all the term that appear only in the document and assign a rank which is the average rank from $p + n + 1$ to $p + m + n$, i.e., $p + n + m/2$. For all the terms that appear only in the category, they are assign a rank which is the average rank from $p + m + 1$ to $p + m + n$, i.e., $p + m + n/2$.
2. **Bigger Set.** In this heuristic, we only consider all the terms from the bigger term collection, which is usually the category. Similarly, for the same example, we assign a rank which is the average rank from $p + m + 1$ to $p + m + n$, i.e., $p + m + n/2$.
3. **Smaller Set.** In this heuristic, we only consider all the terms from the smaller term collection, which is usually the document. Similarly, for the same example, we assign a rank which is the average rank from $p + n + 1$ to $p + m + n$, i.e., $p + n + m/2$.
4. **Intersection.** In this heuristic, we only consider all the terms that appear in both the document and the category. Therefore, we do not need to adjust the rank of any terms.

Two vectors of ranks of the same size will be generated after using any of the above heuristics. The vectors will be directly used as the inputs for the Spearman Rank-Order Correlation Coefficient algorithm. The result measures how similar the document and the category is, and will be used in our k-NN classifier.

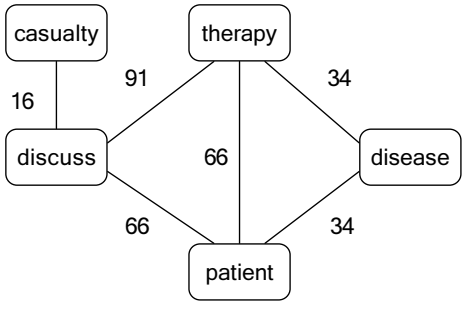
Classification. We adopt the following simple classifier to perform the text classification. We first build a set of vectors of rank values, $\{V_1, V_2, \ldots, V_n\}$, representing the categories $\{C_1, C_2, \ldots, C_n\}$ from the training set. For each testing document, a vector of rank values, F, representing the testing document D is calculated. We search for the category C such that it has the highest rank correlation coefficients with the testing document D. Then the document is assigned to the category C.

4.2 Classification Based on the Term Distances

Another similarity function we propose is based on the intuition that the distance between two terms in the term graph reflects the relations between the terms. Intuitively, terms that appear more often in the text collections will have more chances to be connected directly in the term graph.

Term Distance Matrix. Given a term graph, we can build its term distance matrix as follows. Assume the graph has n terms. Its term distance matrix T is of size $n \times n$, where $T[i][j]$ records the smallest number of hops between term i and j.

Example 2. Consider the term graph shown in Figure 3(a). Its distance matrix is shown in Figure 3(b).



(a) The Term Graph

$$\begin{bmatrix} 0 & 2 & 1 & 2 & 3 \\ 2 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 & 2 \\ 2 & 1 & 1 & 0 & 1 \\ 3 & 1 & 2 & 1 & 0 \end{bmatrix}$$

(b) The Corresponding Distance Matrix

Fig. 3. An Example Term Graph and Its Distance Matrix

Distance-Weighted Similarity. We propose to use the *Distance-Weighted Similarity* to characterize the similarity of a document and a category. Intuitively, if a document D is similar to a category C, there will be many pairs of terms that occur in both the document and the category; in addition, the distance between terms in those pairs will not be too large. We show the algorithm in Algorithm 1. α is a parameter to adjust the effect of the distance of the terms to the similarity score. We set $\alpha = 2$ in our experiments.

Algorithm 1 Distance-Weighted-Similarity(T, D, α)

Input:
T is the distance matrix for the category C.
Description:
1: $n = 0$
2: $w = 0$
3: **for all** pair of terms, (u, v) , in D **do**
4: $w = w + (T[u][v])^{\alpha}$
5: $n = n + 1$
6: **end for**
7: **return** $\frac{n}{w}$

Classification. We adopt the following simple classification method based on the distance matrix and the distance-weight similarity function. Given the set of distance matrixes $\{T_1, T_2, \ldots, T_n\}$ representing the categories $\{C_1, C_2, \ldots, C_n\}$ and a testing document D. The document will be classified to category C_i if and only if the distance-weighted similarity of C_i and D is the largest among all the categories.

5 Experimental Evaluation

In this section, we present some preliminary experiment results using classifiers build on our term graph model. We note that our current focus is to more

Table 1. Statistics of the Categories

Category	Training Set	Testing Set	*minsup*
acq	1488	643	12
corn	159	48	6
crude	349	161	10
earn	2709	1044	15
grain	394	134	8
interest	289	100	12
money-fx	460	141	20
ship	191	85	7
trade	337	112	14
wheat	198	66	6

on gaining more insight into the term graph model and exploring its potential applications in text mining.

We have implement our two classification methods in Java. We choose to use the SVM classifier with linear kernel for comparison, as it is one of the most best text classifier[8]. We build our SVM classifier built using TF-IDF weighting scheme on top of the `libsvm` library [4].

The text collections we used in the experiments are the Reuters-21578 repository, which is the standard collections in many previous studies on text categorization. There exist several modes of splitting the Reuters-21578 text collections into the training and testing parts. For comparison purpose, we choose to use "ModApte" split, which produces 9603 documents for the training set and 3299 documents for the testing set. We also follow the popular approach to choose to use only top-10 categories with the most number of training documents in the experiment [18]. We list the category names, the corresponding numbers of training and testing documents, and the minimum support thresholds in Table 1. The minimum support thresholds are set empirically such that the size of the term graphs for the categories are of similar size.

We measure the adjusted accuracy of different classifiers on the testing documents. The adjustment is necessary because a Reuters document may belong to multiple categories. We regard it as a correct classification as long as the predicted class label match one of the class labels of the testing document.

5.1 Visualization of the Graph Model

We show the plot of an example term graph (with PageRank scores) for a medical text document collections in Figure 4. We observe that many important notions have been captured in the figure, such as "patient" with "disease".

5.2 Experiments Using the Rank-Based Classification

We performed experiments using classification methods based on the notion of rank correlation. We list the results of each of the four heuristics in Table 2.

As shown in the tables above, method of using Union Set is most competitive: it has similar accuracy with SVM for four out of ten categories. Specifically, it

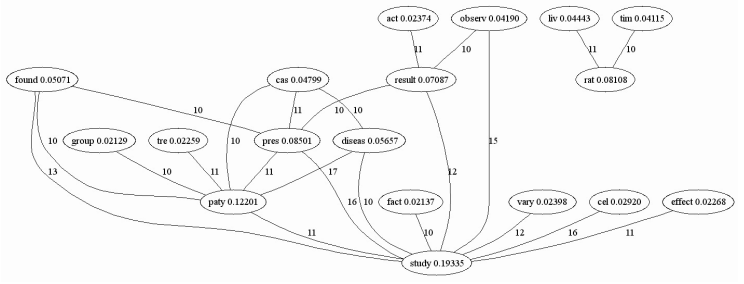


Fig. 4. Visualization of the Term Graph

Table 2. Experiments Using the Rank-based Classification

Category	Union Set	Bigger Set	Smaller Set	Intersection	SVM
acq	98.4	80.9	34.5	20.7	95.6
corn	93.7	70.8	62.5	31.2	93.8
crude	83.2	70.8	65.2	35.4	90.1
earn	95.4	95.3	64.9	27.0	98.8
grain	90.2	51.5	43.2	20.1	94.8
interest	57.0	70.0	55.0	40.0	83.0
money-fx	51.8	75.9	57.4	52.5	90.8
ship	58.8	65.9	58.8	35.3	84.7
trade	80.3	70.5	49.1	27.7	89.3
wheat	96.9	53.0	50.0	21.1	100.0

significantly outperforms SVM for acq category. Considering the simplicity of our classifier, these results are rather encouraging.

We can also observe that the methods of using Intersection Set or Smaller Set do not produce good results. The problem for these methods is that the input vectors are not able to represent the category. There are a large number of words used to present the whole category. For example, the acq category has 776 words, the earn category has 527 words. Using Intersection or Smaller sets means that we use only a small portion of those when we calculate the correlation between the category and the testing document. Therefore, the accuracy is very low. The accuracy of Bigger and Union sets are better for some categories.

5.3 Experiment with Distance Score Approach

We experimented with the classifier based on the distance-weighted similarity function and list the results in Table 3.

We can observe the similar trend between the results in this experiment and the Bigger Set and Union Set methods above. earn, acq have the highest precision points. The system does not perform well for overlapped categories such as grain, corn and wheat because those categories have lots of words in common. Unfortunately, those words also appear frequently in the testing documents and

Table 3. Experiments Using the Distance-Weighted Classification

Category	Distance-Weighted	SVM
acq	87.9	95.6
corn	60.4	93.8
crude	52.8	90.1
earn	88.3	98.8
grain	54.5	94.8
interest	68.0	83.0
money-fx	80.1	90.8
ship	22.4	84.7
trade	73.2	89.3
wheat	63.6	100.0

the distance model cannot clearly discriminate between the categories to which the documents should belong.

6 Conclusions

In this work, we introduce a new term graph model to capture more information for text document and present preliminary results on its potential application in text mining. The new model is capable of capturing the term co-occurrence information among terms. We explored ideas of using novel similarity functions, the rank correlation coefficient and the distance-weighted similarity function, both based on the new model.

There are many area our methods can be improved. As one of our future work, we are actively exploring new features based on our term graph model. Another promising direction is to use our model to complement existing classification method.

References

1. M. Antoine and O.R. Zaiane. Classifying text documents by associating terms with text categories. In *Proceedings of the 13th Australasian conference on database technologies*, volume 5, pages 215–222, Melbourne, Australia, 2002.
2. F. Beil, M. Ester, and X. Xu. Frequent term-based text clustering. In *Proceedings of the 8th International Conference on Knowledge Discovery and Data Mining*, 2002.
3. S. Brin and L. Page. The anatomy of a large-scale hypertextual web search engine. In *Computer Networks and ISDN Systems*, volume 30, pages 107–117, 1998.
4. C. C. Chang and C. J. Lin. LIBSVM: A library for support vector machine, 2001. At `http://www.csie.ntu.edu.tw/~cjlin/libsvm/`.
5. C. Cortes and V. Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, September 1995.
6. B. C. M. Fung, K. Wang, and M. Ester. Hierarchical document clustering using frequent itemsets. In *Proceedings of the SIAM International Conference on Data Mining*, 2003.

7. P. Jackson and I. Moulinier. *Natural Language Processing for Online Applications: Text Retrieval, Extraction and Categorization*. John Benjamins Publishing Company, Amsterdam/Philadenphia, 2002.
8. T. Joachims. Text categorization with support vector machines: Learning with many relevant features. In *Machine Learning: ECML- 98, 10th European Conference on Machine Learning, 1998*, pages 137–142, 1998.
9. B. Liu, C. W. Chin, and H. T. Ng. Mining topic-specific concepts and definitions on the web. In *Proceedings of the 12th International Conference on World Wide Web*, pages 251–260, 2003.
10. Guimei Liu, Hongjun Lu, Jeffrey Xu Yu, Wei Wang, and Xiangye Xiao. AFOPT: An efficient implemetation of pattern growth approach. In *Workshop on Frequent Itemset Mining Implementations*, Melbourne, Florida, USA, November 2003.
11. Chris D. Paice. Another stemmer. *SIGIR Forum*, 24(3):56–61, 1990. `http://www.comp.lancs.ac.uk/computing/research/stemming/`.
12. William H. Press, Brian P. Flannery, Saul A. Teukolsky, and William T. Vetterling. *Numerical Recipes in C: The Art of Scientific Computing*. Cambridge University Press, 2nd edition edition, 1992. ISBN 0-521-43108-5.
13. G. Salton and M. J. McGill. *An Introduction to Modern Information Retrieval*. McGraw-Hill, 1983.
14. F. Sebastiani. Machine learning in automated text categorization. Technical Report Technical Report IEI-B4-31-1999, Consiglio Nazionale delle Ricerche, Pisa, Italy, 1999.
15. Fabrizio Sebastiani. Machine learning in automated text categorization. *ACM Computing Survey*, 34(1):1–47, 2002.
16. K. Wang, C. Xu, and B. Liu. Clustering transactions using large items. In *CIKM 1999*, 1999.
17. Y. Yang. An evaluation of statistical approaches to text categorization. Technical Report Technical Report CMU-CS-97-127, Carnegie Mellon University, April 1997.
18. Y. Yang and X. Liu. A re-examination of text categorization methods. In *22nd Annual International SIGIR-99*, pages 42–49, Berkley, August 1999.