An Intelligent Support System for Developing Text Classifiers

Krisztian Balog
[krisztian@balog.hu]

advisor: dr. Wojtek Kowalczyk

second reader: dr. Elena Marchiori

Vrije Universiteit
Department of Computer Science
Amsterdam, The Netherlands

2004

Abstract

In this paper we introduce a generic text mining system, called `iTM`. `iTM` builds models and supports the classification process. It deals with different kind of sources: text files, e-mail/newsgroup messages and HTML pages. `iTM` provides rich user feedback: important words and phrases of the text which are really significant in decision making. We have implemented a few active learning methods which reduce the number of manually labeled examples.
We demonstrate the usefulness of `iTM` by using it for labeling 13000 HTML pages of the www.cs.vu.nl domain. The full classification process was done in a few hours. With 550 manually labeled examples the accuracy was given to 74%.
Moreover we run a number of experiments on the 20-newsgroups dataset. We demonstrate that by using our system the number of manually labeled documents can be decreased by 45% to achieve 80% accuracy. Furthermore, with only 100 labeled examples our methods reach 42% accuracy.

Keywords and Phrases: textmining, Naive Bayes, Boostexter, supervised learning, intelligent sample selection, use of unlabeled data, active learning, prior user knowledge, user feedback

Acknowledgments

My gratitude goes first to my advisor, dr.Wojtek Kowalczyk for his generous support and guiding of my work.
I am thankful for the opportunity to be given by the Vrije Universiteit Amsterdam to be an exchange student during the 2003 fall and 2004 spring semesters. I am grateful to all the nice people I met in the Netherlands.
Many thanks to Peter Hofgesang and Maria Joosz, who were not just colleagues but friends as they helped and encouraged me during my stay.
At last but not least I am glad that I have a beautiful girlfriend who was highly patient with me and let me go to study many miles from home and let me complete this work. For this and for all: Thank you Noemi!

Contents

Chapter 1
Introduction

1.1 Motivation

Text categorization or *classification* is the problem of classifying text documents into categories or classes. Documents can be any kind of text files: e-mails, newsletters or html pages. Classification can be based on several viewpoints: topic, style, language, etc.
In *automated text classification* we need to attach labels to a set of documents, called the *traning examples*. Then a model is built based on these examples and the remaining part of the data set is labeled. It means that the classification algorithm predicts the target class labels for each document with a score or probability.
Labeling the training examples is a costly operation since one has to read each of these documents to obtain the target class. The motive of the project is to reduce effort on example selection. To achieve this, we use and combine recent machine learning methods and find out new ones as well.

Existing statistical text learning algorithms can be trained to approximately classify documents, given a sufficient set of labeled training examples. These text classification algorithms have been used to automatically catalog news articles ([4], [22]) and web pages([12]), automatically learn the reading interest of users ([15], [10]), and automatically sort electronic mail ([3]).
One key difficulty with these current algorithms, and this issue addressed by this paper, is that they require a large number of training examples to learn accurately. Labeling must often be done by a person; this is a painfully time-consuming process. Most users of a practical system would not have patience to label thousands of documents. One would obviously prefer algorithms that can provide accurate classification after hand-labeling only a few dozen instances.
This paper addresses the problem of learning accurate text classifiers from limited number of labeled examples. Over the course of several experimental comparisons, we show that (1) reducing the size of the vocabulary can significantly increase performance, (2) using the richer structure of e-mail messages/html pages improves classification accuracy up to 8%, (3) k-means clustering is a right alternative for selecting initial samples, (4) use of unlabeled data improves classification accuracy, (5) intelligent sample selection achieves 18% reduction in classification error, and (6) using prior user knowledge can improve classification accuracy up to 12%. We also describe a method for producing rich feedback to users.

The above described features are implemented and integrated into a visual system, called `iTM` (intelligent Text Mining). `iTM` supports the process of text categorization and provides rich feedback to users. It is written in Java using J2SDK thus it is platform independent: it runs under Windows, Linux or Solaris. The code conforms to the GNU coding standards and the full source is available under the GNU Public License. The system is stand-alone, but designed to be expandable, thus it accepts enlargements as new classification algorithms, new types of inputs, etc.
We have tested `iTM` on the web pages of the Computer Science Department of Vrije Universiteit, Amsterdam. 13000 html documents were classified with 74% accuracy by labeling only 550 pages. The full process was done in a few hours.

1.2 Paper outline

The following chapter defines the basic concepts of text mining as well as notation, data representation and classification algorithms. Chapter 3 presents the elements of the intelligent support system. These are machine learning methods which can improve the performance: intelligent sample selection, active learning, use of unlabeled data, use of prior user knowledge and deep feedback. Chapter 4 presents the overview of the system. Chapter 5 contains the results of some experiments on different data sets. Finally, Chapter 6 presents the paper's main conclusions.

Chapter 2
Text mining

In this chapter we introduce and formalize the problem of text classification. We present the process of text mining and deal with the representation of textual documents. The basic ideas of two classification algorithms (Naive Bayes, Boostexter) are described here. We also present a possible way of implementation, with respect to practical difficulties.

2.1 Text classification problem

Text categorization or *classification* is the problem of classifying text documents into categories or classes. Documents can be any kind of text files: e-mails, newsletters or html pages. Classification can be based on several viewpoints: topic, style, language, etc. Text classification problems are usually *multiclass*, which means that there are two or more possible categories. In this work we focus on moderate number of categories.
Classifying a document means that we attach a class label to it. Text-categorization tasks can be *multi-labeled* in the sense that the same document may be relevant to more than one category. While most machine-learning systems are designed to handle multiclass data, much less common that systems can handle multi-label data. In this work we also focus on *single labeled* data, but we give a possible solution for handling multi-labels in the corresponding chapter.
In our approach we are trying to separate a given set of documents based on their content. For that we need to build a classifier using some training examples. It means that we assign a label to these samples and then the classifier predicts the label of each document of the remaining set. The detailed description of the process is presented in the following section.
A typical text classification problem is for instance to classify news articles by topic based on their textual content. We have a given number of newsgroups and we train a classifier by labeling a set of these news documents. Then a classifier can automatically predict the topic of a new instance by examining its content. In these real-world tasks it is expensive to acquire a sufficient number of labeled examples for training. This paper investigates methods for reducing the cost of sample selection.

Hereunder we define it formally:
Let $\mathcal{D}$ denote a domain of possible text documents and let C be a finite set of labels (target classes). In the traditional machine-learning setting, each document $d \in \mathcal{D}$

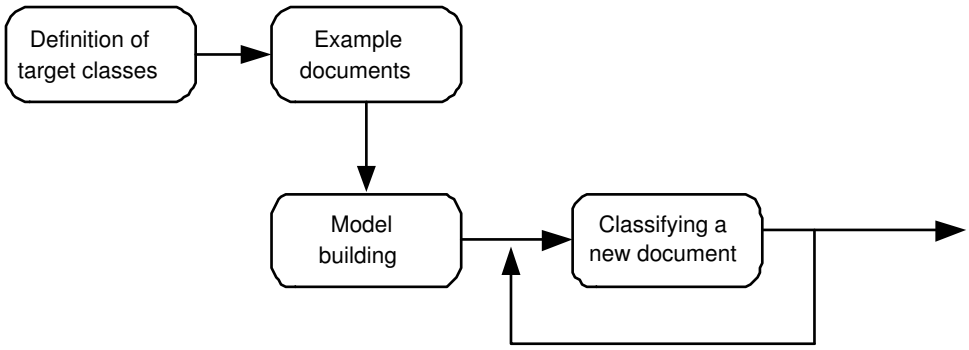

Figure 2.1: Text mining process

is assigned to one single class $c \in C$. We are given some training examples $S \subset \mathcal{D}$. Then the aim is to find a classifier $H : \mathcal{D} \to C$ which minimizes the probability that $c \neq H(D)$ on a newly observed document (d, c).
In the multi-label case, each document $d \in \mathcal{D}$ may be assigned multiple labels in C. (For example, in a multi-label news-filtering problem, a document may belong to both Finance and Sport). Thus, a labeled example is a pair $(d, \mathcal{C})$ where $\mathcal{C} \subseteq C$ is the set of labels assigned to d. The single-label case is a special case in which $|C| = 1$.

2.2 Text mining process

The *text mining process* begins with the definition of target classes. It simply means that we enumerate the possible categories (for example: Sport, Finance, Weather,...). As we will see in Section 4.5 it is not as trivial when we allow users to add/remove categories real-time, during the classification process.
To build a classification model we label some example documents. Note that it is a costly operation since the user has to read the text to assign the proper label.
The classification model is built based on the training examples. Finally, any number of new documents can be classified using this model. A new document's classification means that we determine the target class, it most probably belongs. We also provide a likelihood for each target class. Apparently, the most probable is picked for estimated label.
A frequent measure of a classification model is the ratio of properly classified documents.

2.3 Representation

Documents can be represented in many different ways, but usually it is preceded with some *pre-processing*. Pre-processing means that we remove the punctuation marks, spaces, end of line characters and make all words lowercase. To use the richer structure of formatted documents (html pages for instance) is also object of

word	occurrence
the	7
text	1
feature	1
case	2
contain	1
average	2
...	

Table 2.1: Example of bag-of-words representation using frequency vector

this work.

2.3.1 Text representation

One well-established technique for text classification is to represent each document as a bag-of-words. In this representation, each text document is represented as a vector of numbers, each component corresponds to the count of a particular word in the document. No ordering of words or any structure of text is used. (See Table 2.1).
We have mentioned pre-processing already, but usually there is some more to do, before we store documents in this representation. Words that occur just a few times or have "no meaning" (and, or, a, the, ...) are not just unnecessary, but cause inconvenience to classification.
It is a well accepted solution that a *vocabulary* is built which contains all different words and frequencies from the training examples. Then the less frequent words are removed as well as the *stop words*. Stop words are the most frequent ones, usually: and, or, not, of, ... As we will see later there are more other reasons, why to maintain a vocabulary with acceptable size (means a few thousand words).
Then a document is represented with the words appearing in the vocabulary: $\forall d \in \mathcal{D}$: $d = [a_1, ...a_{|V|}]$ where $\mathcal{D}$ denotes the space of all possible text documents and $|V|$ refers to the size of the vocabulary. a_i denotes the frequency of the ith word of the vocabulary in document d.

2.3.2 Representation of HTML documents

The previously described approach can be applied to all textual documents, but then we may lose the information gained from formatting. In order to use the richer structure of HTML pages we need to consider the representation of HTML tags, such as title, hyperlinks, headings, search keywords.
A possible solution can be to build a classifier based on these tags, like in [19] but the benefits of this approach have not been fully realized yet.
We could enrich the bag-of-words representation by adding weights to the HTML tags. It simply means that if a word appears in a given HTML tag, then we count it not once, but corresponding to its weight (1,2,3,...). Weights can be stored in a

tag	weight
title	4
a href	2
h1	3
h2	2
...	

Table 2.2: Example of HTML tags' weights

table, like in Table 2.2. The advantages of this approach:

- weights can be tuned according to experiments
- optional if we take a tag into consideration
- works for other documents where we want to pick out special or discriminating parts (e.g. e-mail's title)

2.4 Classification algorithms

In this section we introduce two text classifier algorithms, namely *Naive Bayes* and *Boostexter*. The `iTM` system includes the implementation of these, but also prepared to accept other algorithms as enlargements.

2.4.1 Naive Bayes

The general algorithm presented here for learning to classify text, based on the naive Bayes classifier. Probabilistic approaches such as naive Bayes described here are among the most effective algorithms currently used for learning to classify text documents. The following representation is based on the *Bayesian learning* chapter of Tom Mitchell's Machine Learning book ([13]).

One highly practical Bayesian learning method is the naive Bayes learner, often called the *naive Bayes classifier*. It has been shown ([13], [21]) that in some domains its performance is comparable to neutral network and decision tree learning. The naive Bayes classifier applies to learning tasks where each instance x is described by a conjunction of attribute values and where the target function $f(x)$ can take on any value from some finite set C. A set of training examples of the target function is provided, and a new instance is presented, described by the tuple of attribute values $< a_1, a_2, ..., a_n >$. The learner is asked to predict the target value, or classification, for this new instance.
In our case instances are text documents. C denotes the set of categories, while $c_i \in C$ denoting the ith target class.
The Bayesian approach to classifying the new instance is to estimate $P(c_i|a_1, a_2, ..., a_n)$ for $\forall c_i \in C$, more precisely find c_j that maximizes $P(c_j|a_1, a_2, ..., a_n)$. In other words we assign the most probable target value, c_{MAP}, given the attribute values $< a_1, a_2, ..., a_n >$ that describe the instance.

$$c_{MAP} = argmax_{c_j \epsilon C} P(c_j|a_1, a_2, ..., a_n) \tag{2.1}$$

Note that (2.1) is not feasible in this form. Bayes theorem states that

$$P(h|D) = \frac{P(D|h)P(h)}{P(D)} \tag{2.2}$$

Using Bayes theorem to rewrite the expression (2.1), we get

$$\begin{aligned} c_{MAP} &= argmax_{c_j \epsilon C} \frac{P(a_1, a_2, ..., a_n|c_j)P(c_j)}{P(a_1, a_2, ..., a_n)} \\ &= c_{MAP} = argmax_{c_j \epsilon C} P(a_1, a_2, ..., a_n|c_j)P(c_j) \end{aligned} \tag{2.3}$$

In the final step above we dropped the term $P(a_1, a_2, ..., a_n)$ because it is a constant independent of c_j. Now it is possible to estimate the two terms in Equation (2.3) based on the training data. It is easy to estimate each of the $P(c_j)$ values simply by counting the frequency with which each target value c_j occurs in the training data. However, estimating the different $P(a_1, a_2, ..., a_n|c_j)$ terms in this form is not

manageable unless we have a very large set of training data (the problem is that the number of these terms is equal to the number of possible instances times the number of possible target values. Therefore, we need to see every instance in the instance space many times in order to obtain reliable estimates).
The naive Bayes classifier is based on the simplifying assumption that the attribute values are conditionally independent given the target value. It means that the probability of observing the conjunction $a_1, a_2, ..., a_n$ is just the product of the probabilities for the individual attributes: $P(a_1, a_2, ..., a_n|c_j) = \prod_i P(a_i|c_j)$. Substituting this into Equation (2.3), we have the approach used by the naive Bayes classifier.

Naive Bayes classifier:

$$c_{NB} = argmax_{c_j \epsilon C} P(c_j) \prod_i P(a_i|c_j) \tag{2.4}$$

where c_{NB} is the target value output by the naive Bayes classifier. Notice that in a naive Bayes classifier the number of distinct $P(a_i|c_j)$ terms that must be estimated from the training data is just the number of distinct attribute values times the number of distinct target values, which is a much smaller number than if we were to estimate the $P(a_1, a_2, ..., a_n|c_j)$ terms.
To summarize, the naive Bayes learning method involves a learning step in which the various $P(c_j)$ and $P(a_i|c_j)$ terms are estimated, based on their frequencies over the training data. The set of these estimates corresponds to the learned hypothesis. This hypothesis is then used to classify each new instance by applying the rule in Equation (2.4).

How to classify text

The two main design issues involved when applying the naive Bayes classifier to such text classification problems are (1) decide how to represent an arbitrary text document in a terms of attribute values, and (2) to decide how estimate the probabilities required by the naive Bayes classifier.
Given a text document, we define an attribute for each word position in the document and define the value of that attribute to be the English word found in that position. The following sentence "This is an example sentence" would be described by 5 attribute values, corresponding to the 5 word positions. The value of the first attribute is the word "*this*", the value of the second attribute is the word "*is*", and so on. Apparently, long text documents will require a larger number of attributes than short documents. (As we shall see, this will not cause us any trouble).
Given this representation for text documents, we can now apply the naive Bayes classifier. Let us assume we are given a set of training documents classified as *like* or *dislike* and we are now given a new document and asked to classify it. For the sake of concreteness let us assume the new text document is the example sentence: "This is an example sentence". In this case we instantiate Equation (2.4) to calculate

the naive Bayes classification as

$$c_{NB} = argmax_{c_j \epsilon \{like, dislike\}} P(c_j) \prod_{1}^{5} P(a_i|c_j)$$
$$= argmax_{c_j \epsilon \{like, dislike\}} P(c_j) P(a_1 = "this"|c_j)$$
$$P(a_2 = "is"|c_j)...P(a_5 = "sentence"|c_j)$$

To calculate c_{NB} using the above expression, we require estimates for the probability terms $P(c_j)$ and $P(a_i = w_k|c_j)$ (w_k is the kth word in the English vocabulary). The first can be easily estimated based on the fraction of each class in the training data (eg. $P(like) = 0.4$ and $P(dislike) = 0.6$). Estimating the class conditional probabilities (eg. $P(a_1 = "this"|like)$) is more problematic because we must estimate one such probability term for each combination of text position, English word, and target value. (Approximately 50.000 distinct words in the English vocabulary, 2 possible target values, 5 text positions in the current example: $2 \cdot 5 \cdot 50.000 \approx 500.000$ such terms from the training data - and usually documents are longer than 5 characters and we may want to use more than two target classes).
The naive Bayes classification c_{NB} is the classification that maximizes the probability of observing the words that were actually found in the document, subject to the usual naive Bayes independence assumption. The independence assumption [1] (2.4) states in this setting that the word probabilities for one text position are independent of the words that occur in other positions, given the document classification c_j. This amounts to assuming that the attributes are independent and identically distributed, given the target classification.
Formally, $P(a_i = w_k|c_j) = P(a_m = w_k|c_j)$ for all i, j, k, m. Therefore we estimate the entire set of probabilities $P(a_1 = w_k|c_j), P(a_2 = w_k|cj), ...$ by the single position-independent probability $P(w_k|c_j)$, which we will use regardless of word position. Now we require only $|C| \cdot |Vocabulary|$ distinct terms of the form $P(w_k|c_j)$, what is still a large number, but manageable.

To complete the design of our learning algorithm, we must still choose a method for estimating the probability terms. The most logical choice would be $\frac{n_c}{n}$, where n is the total number of training examples, n_c is a number of these for which the condition is true. It provides a good estimate of the probability in many cases, but it provides poor estimates when n is very small.
To see the problem, imagine that we have the value of $P(w_k = "word"|c_j = "dislike")$ is 0.06 and we have a sample of containing only 5 examples for which $c_j = "dislike"$. Then the most probable value for n_c is 0. It leads to two difficulties. First, $\frac{n_c}{n}$ produces a biased underestimate of the probability. Second, when this probability estimate is zero, this probability term will dominate the Bayes classifier if the future query contains $w_k = "word"$. The reason is that the quantity calculated in Equation (2.4) requires multiplying all the other probability terms by this zero

[1] Note this assumption is clearly incorrect. For example, the probability of observing the word "learning" in some position may be greater if the preceding word is "machine". Despite this obvious inaccuracy of the independence assumption, in practice the naive Bayes learner performs remarkably well in many text classification problems despite the incorrectness of this independence assumption.

value.
To avoid this difficulty we adopt a Bayesian approach to estimating the probability, using the m-estimate defined We adopt the m-estimate method with uniform priors and with m equal to the size of the word vocabulary.
m-estimate of probability:

$$\frac{n_c + mp}{n + m} \tag{2.5}$$

where p is our prior estimate of the probability we wish to determine, and m is a constant called the *equivalent sample size*, which determines how heavily to weight p relative to the observed data. A typical method for choosing p in the absence of information is to assume uniform priors: if an attribute has k possible values we set $p = \frac{1}{k}$. Note, if m is zero, the m-estimate is equivalent to simple fraction $\frac{n_c}{n}$. (m is called equivalent sample size because Equation (2.5) can be interpreted as augmenting the n actual observations by an additional m virtual samples distributed according to p).

Thus, the estimate for $P(w_k|c_j)$ will be

$$\frac{n_k + 1}{n + |Vocabulary|} \tag{2.6}$$

where n is the total number of word positions in all training examples whose target value is c_j and n_k is the number of times word w_k is found among these n word positions, and $|Vocabulary|$ is the total number of distinct words found within the training data.

The following algorithm is implemented for learning and classifying text documents. In addition to the usual naive Bayes assumptions, these algorithms assume the probability of a word occurring is independent of its position within the text.
The implementation used a particular simplification too: only a subset of the words occurring in the documents were included as the value of the $Vocabulary$ variable in the algorithm. Words with very low and very high frequency were removed (see Chapter 4.3).

LEARN-NAIVE-BAYES-TEXT($Examples, C$)

Examples is a set of text documents along with their target values. C is the set of all possible target values. This function learns the probability terms $P(w_k|c_j)$, describing the probability that randomly drawn from a document in class c_j will be the English word w_k. It also learns the class prior probabilities $P(c_j)$.

1. create $Vocabulary$ by collecting all words that occur in $Examples$
2. calculate the required $P(c_j)$ and $P(w_k|c_j)$ probability terms
   - For each target value c_j in C do

- $D_j \leftarrow$ the subset of documents from $Examples$ for which the target value is c_j
- $P(c_j) \leftarrow \frac{|D_j|}{|Examples|}$
- $n \leftarrow$ total number of distinct word positions in D_j (sum of the length of $\forall d \in D_j$)
- for each word w_k in $Vocabulary$
  - $n_k \leftarrow$ number of times word w_k occurs in D_j
  - $P(w_k|v_j) \leftarrow \frac{n_k+1}{n+|Vocabulary|}$

CLASSIFY-NAIVE-BAYES-TEXT(d)

Return the estimated target value for the document d. a_i denotes the word found in the ith position within d.

- *positions* $\leftarrow$ all word positions in d that contain words found in $Vocabulary$
- Return c_{NB} where $c_{NB} = argmax_{c_j \in C} P(c_j) \prod_{i \in positions} P(a_i|c_j)$

During learning, the procedure LEARN-NAIVE-BAYES-TEXT examines all training documents to extract the vocabulary of all words and tokens that appear in the text, then counts their frequencies among the different target classes to obtain the necessary probability estimates. Later, given a new document to be classified, the procedure CLASSIFY-NAIVE-BAYES-TEXT uses the probability estimates to calculate c_{NB} according to Equation (2.4). (Note that any words appearing in the new document that were not observed in the training set are simply ignored).

Implementation

This section describes a possible way of implementation of the naive Bayes classifier. We store all documents in a two-dimensional matrix where an element in the ith row and jth column gives how many times the jth word of the vocabulary appears in the ith document (See Table 2.3).
This matrix is being created at the same time with the document's loading and parsing, and it contains all distinct words occurring in the set of documents. It has two reasons. First, at the beginning it is unknown which documents are going to be the initial examples. Second, the goal of this system to give help to the user to label just the most relevant documents (and just as many as necessary). To choose these documents smartly, we need to have all documents in the memory before the classification procedure starts.

After having the initial examples selected, we need to calculate (Table 2.4):

- for each target and each word of the vocabulary, how many times a given word occurs in the examples of that target (#examples)

words/documents	any	word	example	document	also	these	...
document 1	0	2	0	0	0	4	...
document 2	1	0	1	2	0	0	...
document 3	0	3	1	1	0	2	...
...	...						

Table 2.3: Representation of documents in a two-dimensional matrix

words/documents	any	word	example	also	these	...	target	
document 1	0	2	0	0	4	...	1	
document 2	1	0	1	0	0	...	2	
document 3	0	3	1	0	2	...	2	
...	...							
target	any	word	example	also	these	...	#examples	#positions
target v_1	0	2	0	0	4	...	1	113
target v_2	1	3	2	0	2	...	2	270
...	...							

Table 2.4: Representation of target classes

- the total number of distinct word positions for that target(#positions), which is equal to the sum of all word frequencies occurring in the target's row

The calculation of the naive Bayes probabilities goes according to the pseudo code in Figure 2.2. To obtain a new document's (d) estimated target value we use the algorithm described in Figure 2.3. ww_k denotes the kth word of d, $besttarget$ contains the estimated target value for d and $bestval$ means the estimated probability for this target.

As we might see, it works properly, except one thing: the probabilities we have to multiply by each other are really small numbers ($< 10^{-3}$ for a vocabulary with only a few thousand words), therefore all estimation values will be close to 0 and their products in practice is 0 (due to limited precision in arithmetic). One possible solution of this problem to use lead log and $\sum$ in, instead of $\prod$.

In case of two target classes, we calculate the following for a new document to

for $j = 1$ to $|C|$
    $P(c_j) = \frac{target[j][\#examples]}{sum\#examples}$;
    for $k = 1$ to $|Vocabulary|$
        $P(w_k|c_j) = \frac{target[j][k]+1}{target[j][\#positions]+|Vocabulary|}$;
    end_for(k)
end_for(j)

Figure 2.2: Pseudo code of Naive Bayes probability calculation

```
bestval=0;
besttarget=0;
for j=1 to |C|
    thisval=P(c_j);
    for k=1 to |words in d|
        thisval=thisval*P(ww_k|c_j);
    end_for(k)
    if thisval>bestval ⇒ bestval=thisval;
                         besttarget=j;
end_for(j)
```

Figure 2.3: Pseudo code for obtaining a new document's estimated target value with Naive Bayes

be classified:

$$P = P(c_1) \prod_{i \epsilon positions} P(a_i|c_1) \tag{2.7}$$

$$Q = P(c_2) \prod_{i \epsilon positions} P(a_i|c_2) \tag{2.8}$$

We know that

$$\frac{P}{\alpha} + \frac{Q}{\alpha} = 1 \Rightarrow P + Q = \alpha \tag{2.9}$$

where $\alpha = P(a_1, a_2, ..., a_n)$ (See Equation(2.3)).
To determine real probabilities, we are interested in $\frac{P}{\alpha}$ and $\frac{Q}{\alpha}$.

Let us initialize two new variables:

$$p = \log P \Rightarrow P = 10^p \tag{2.10}$$

$$q = \log Q \Rightarrow Q = 10^q \tag{2.11}$$

where p is calculated in the following way:

$$\begin{aligned} p = \log P &= \log(P(c_1) \prod_{i \in positions} P(a_i|c_1)) \\ &= \log P(c_1) + \log(\prod_{i \in positions} P(a_i|c_1)) \\ &= \log P(c_1) + \sum \log(P(a_i|c_1)) \end{aligned} \tag{2.12}$$

Our goal is to get

$$\frac{P}{\alpha} = \frac{P}{P+Q} = \frac{10^p}{10^p + 10^q} \tag{2.13}$$

(2.13) cannot be calculated directly, because at this point we would meet with the same problem than previously (Experimentally $p, q < 10^{-3}$, therefore calculating (2.13) generates 0 in practice due to limited precision in arithmetic). Let us define

a the following way:

$$a = -\frac{p+q}{2} \tag{2.14}$$

Multiplying (2.13) by 10^a gives the formula:

$$\frac{10^p \cdot 10^a}{(10^p + 10^q) \cdot 10^a} = \frac{10^{p+a}}{10^{p+a} + 10^{q+a}} \tag{2.15}$$

Using (2.15) makes us possible to calculate real probability values ($\frac{P}{\alpha}$ and $\frac{Q}{\alpha}$ where $\frac{P}{\alpha} + \frac{Q}{\alpha} = 1$).

These calculations were for two target classes but can be easily generalized in the following way for m classes:

$$p_j = \log P(c_j) + \sum_{i \epsilon positions} \log P(a_i|c_j) \quad (1 \leq j \leq m) \tag{2.16}$$

$$a = -\frac{p_1 + ... + p_m}{m} \tag{2.17}$$

$$\frac{P_j}{\alpha} = \frac{10^{p_j+a}}{10^{p_1+a} + ... + 10^{p_m+a}} \quad (1 \leq j \leq m) \tag{2.18}$$

According to these, to obtain a new document's (d) estimated target value we use the modified algorithm: (Figure 2.4)

Handling multi-labeled data:
This algorithm can be easily expanded to handle multi-label data, by simply repeating each document once for each of its assigned labels. Since it is designed for single labeled data we have to deal with this operation's cost or choose an algorithm which is proposed to handle multi-labels, like Boostexter (see the following section).

```
a=0;
for j=1 to |C|
    p[j]=log P(c_j);
    for k=1 to |words in d|
        p[j]=p[j]+log P(ww_k|c_j);
    end_for(k)
    a=a+p[j];
end_for(j)
a=-a/|C|;
expsum=0;
for j=1 to |C|
    expsum=expsum+10^(p[j]+a);
end_for(j)
bestval=0;
besttarget=0;
for j=1 to |C|
    thisval=10^(p[j]+a)/expsum;
    if thisval>bestval ⇒ bestval=thisval;
                         besttarget=j;
end_for(j)
```

Figure 2.4: Pseudo code for obtaining a new document's estimated target value with Naive Bayes, using *log* for probability estimation

2.4.2 Boostexter

The algorithm described here is based on the article [21]. Schapire & Singer discussed four versions of this boosting algorithm, one among them, namely the *real AdaBoost.MH* is expressed in this section.

The purpose of boosting is to find many *weak* or *base hypotheses*, then combine them into a single rule, called the *final* or *combined hypothesis*. A separate procedure called the *weak learner* or *weak learning algorithm* is responsible for combining the weak hypotheses.

We use the notation system described in Section 2.1, moreover we define $Y[l]$ for $Y \subseteq C$, $l \in C$ to be the following:

$$Y[l] = \begin{cases} +1, & \text{if } l \in Y \\ -1, & \text{if } l \notin Y \end{cases}$$

Since AdaBoost is designed to handle multi-labels, each document is a pair of (d, Y), where $d \in \mathcal{D}$ and $Y \subseteq C$ is the set of labels attached to d.

The goal of AdaBoost's learning is to produce a score for all possible labels for a given document in that manner that the appropriate labels will appear at the top of the ranking. Formally, the aim is to produce a function $f : \mathcal{D} \times C \to \mathbb{R}$ with the interpretation that, for a given instance d the labels in C should be ordered accord-

Given: $S =< (d_1, Y_1), ..., (d_m, Y_m) >$ where $d_i \in \mathcal{D}$ and $Y_i \subseteq C$
Initialize $D_1(i, l) = \frac{1}{m \cdot |C|}$

For $t = 1, ..., T$:

- Pass distribution D_t to weak learner
- Get weak hypothesis $h_t : \mathcal{D} \times C \to \mathbb{R}$
- Choose $\alpha_t \in \mathbb{R}$
- Update:

$$D_{t+1}(i, l) = \frac{D_t(i, l) exp(-\alpha_t Y_i[l] h_t(d_i, l))}{Z_t}$$

  where Z_t is a normalization factor (chosen so that D_{t+1} will be a distribution)

Output the final hypothesis:

$$f(d, l) = \sum_{t=1}^{T} \alpha_t h_t(d, l)$$

Figure 2.5: The algorithm of AdaBoost.MH

ing to $f(d, \cdot)$. A label l_1 is considered to be ranked higher than l_2 if $f(d, l_1) > f(d, l_2)$.

Let S be a sequence of training examples $< (d_1, Y_1), ..., (d_m, Y_m) >$ where $d_i \in \mathcal{D}$ and $Y_i \subseteq C$. *AdaBoost.MH* maintains a set of weights as a distribution D_t over examples and labels. Initially, this distribution is uniform. On each round, the distribution D_t and the training sequence S is passed to the weak learner who computes a weak hypothesis $h_t : \mathcal{D} \times C \to \mathbb{R}$. We interpret the sign of $h_t(d, l)$ as a prediction as to whether the label l is or is not assigned to d (actually, this is a prediction of the value of $Y[l]$). The magnitude of the prediction $|h(d, l)|$ is interpreted as a measure of confidence.
A parameter α_t is then chosen and the distribution D_t is updated. In the typical case that α_t is positive, the distribution D_t is updated in a manner that increases the weight of example-label pairs which are misclassified by h_t (for which $Y_i[l]$ and $h_t(d_i, l)$ differ in sign). The final hypothesis ranks documents using a weighted vote of the weak hypotheses.

So far we have left unspecified the actual form and implementation of the weak learner, as well as the choice of the parameter α_t. The weak hypotheses can be imagined as one-level decision trees. The test at the root of this tree is a simple check for the presence or absence of a term in a given document. Based on the outcome of this test, the weak hypothesis outputs predictions and confidences that each label is associated with the document.
Formally, w denotes a term, and $w \in d$ means that w occurs in document d. h_w is

a weak hypotheses which make predictions of the form:

$$h_w(d,l) = \begin{cases} c_{0l}, & \text{if } w \notin d \\ c_{1l}, & \text{in } w \in d \end{cases}$$

where c_{0l} and c_{1l} are real numbers [2].
In *AdaBoost.MH with real-valued predictions (real AdaBoost.MH)* c_{jl} ($j \in \{0,1\}$) are unrestricted real-valued predictions and calculated as follows for a given term w: Let $X_0 = \{d : w \notin d\}$ and $X_1 = \{d : w \in d\}$ (in words: X_0 is a set of documents which are not containing the term w, X_1 is just the opposite). Given the current distribution D_t, we calculate the following for each possible label l, for $j \in \{0,1\}$, and for $b \in \{-1,+1\}$:

$$W_b^{jl} = \sum_{i=1}^{m} D_t(i,l)\chi(d_i \in X_j \wedge Y_i[l] = b) \tag{2.19}$$

where

$$\chi(\pi) = \begin{cases} 1, & \text{if } \pi \text{ holds} \\ 0, & \text{otherwise} \end{cases}$$

For readability, we write W_+^{jl} instead of W_{+1}^{jl} and W_-^{jl} instead of W_{-1}^{jl}. In words, W_+^{jl} is the weight of the documents (with respect to the distribution D_t) which are in partition X_j and labeled by l. W_-^{jl} is the weight of the documents which are in partition X_j and not labeled by l.
Choosing

$$c_{jl} = \frac{1}{2}\ln(\frac{W_+^{jl}}{W_-^{jl}}) \tag{2.20}$$

and setting $\alpha_t = 1$ imply that

$$Z_t = 2\sum_{j\in\{0,1\}}\sum_{l\in\mathcal{Y}}\sqrt{W_+^{jl}W_-^{jl}} \tag{2.21}$$

The term with the smallest value of Z_t is chosen (for being weak hypothesis h_t in the current iteration - see Figure 2.5).
It may well happen that W_-^{jl} or W_+^{jl} is very small or even zero, in which case c_{jl} will be very large or infinite. In practice it may cause numerical problems. For this purpose "smoothed" values can be used:

$$c_{jl} = \frac{1}{2}\ln(\frac{W_+^{jl}+\varepsilon}{W_-^{jl}+\varepsilon}) \tag{2.22}$$

[2]Note that c_{0l} and c_{1l} are real numbers here, and not equal to $c_i \in C$ which denotes a class label

	D_1	D_2	...	$D_{\|C\|}$
document 1			...	
document 2			...	
document 3			...	

Table 2.5: Representation of distribution

where

$$\varepsilon = \frac{1}{m \cdot |C|}$$

Since both W_-^{jl} and W_+^{jl} are bounded between 0 and 1, this has the effect of bounding $|c_{jl}|$ by roughly $\frac{1}{2}\ln(1/\varepsilon)$.

Implementation:

The implementation of Boostexter is the following. We have our documents represented in a two-dimensional matrix, the same way as in Naive Bayes (Table 2.3). The distribution D is stored similarly (see Table 2.5).

After the initialization we choose the word with the smallest Z_t value as a weak classifier in each iteration. To calculate Z_t, we need the value of $W_b^{jl}(b \in \{-1,+1\}, j \in \{0,1\}$. Practically it is represented by four vectors (W^0_{-1},W^1_{-1},W^0_{+1},W^1_{+1}), with the length of $|C|$. Each element of the vector refers to a label $l \in C$. c_{0l} and c_{1l} is calculated only for the best found word in each iteration t, but have to be stored for $\forall t \in T$ (it is needed for constructing the final hypothesis). D is updated after each iteration. Note that D is a distribution, but (in this implementation) $\sum(D)$ not absolutely has to be 1, however $sum(D) = W^0_{-1} + W^1_{-1} + W^0_{+1} + W^1_{+1}$.

Taking it all around, Figure 2.6 shows the pseudo code of model building. As we shall see, there are two parameters of the model building which can be tuned: T is a number of iterations, which is actually the number of weak classifiers. The original Boostexter algorithm allows a word to be selected as a weak classifier several times. Experiments confirm that a higher accuracy can be achieved by using a given word only once as a (weak) classifier. More generally, we say that each word can be selected K times. Apparently, the previous condition corresponds to $K = 1$.
Figure 2.7 contains the pseudo code which is used for classifying a new document. f is calculated for each label l, then f_l with the highest value should be selected as the estimated target class.

$\forall i \in \{1..m\} \forall l \in \{1..|C|\}$: D[i][l] = $\frac{1}{m|C|}$;
$\forall w \in \{1..|Vocabulary|\}$: word_used[w] = 0;

for t=1 to T

bestZ = INFINITE;
bestWord = -1;

for w=1 to $|Vocabulary|$
$\forall l \in \{1..|C|\} \forall b \in \{0,1\} \forall j \in \{0,1\}$: W[j][b][l] = 0;
for i=1 to m
if $w \notin d_i \Rightarrow$ j=0;
if $w \in d_i \Rightarrow$ j=1;
for l=1 to $|C|$
if $l \notin Y_i \Rightarrow$ b=0;
if $l \in Y_i \Rightarrow$ b=1;
W[j][b][l] += D[i][l];
end_for(l)
end_for(i)

thisZ = 0;
for i=1 to m
thisZ += 2*$\sqrt{W[0][1][i] \cdot W[0][0][i]} + \sqrt{W[1][1][i] \cdot W[1][0][i]}$;
end_for(i)
if (thisZ<bestZ
$\wedge$ word_used[w]<K) $\Rightarrow$ bestZ = thisZ;
bestWord = w;
bestW = W;
end_for(w)

word_used[bestWord]++;
h_word[t] = words[bestWord];
for l=1 to $|C|$
c0[t][l] = 0.5*log((bestW[0][1][l]+ε)/(bestW[0][0][l]+ε));
c1[t][l] = 0.5*log((bestW[1][1][l]+ε)/(bestW[1][0][l]+ε));
for i=1 to m
if (words[bestWord] $\in d_i$) $\Rightarrow$ h[t][i][l] = c1[t][l];
if (words[bestWord] $\notin d_i$) $\Rightarrow$ h[t][i][l] = c0[t][l];
end_for(i)
end_for(l)

for i=1 to m
for l=1 to $|C|$
if $l \in Y_i \Rightarrow$ y = 1;
if $l \notin Y_i \Rightarrow$ y = -1;
D[i][l] = D[i][l]*exp(-y*h[t][i][l])/bestZ;
end_for(l)
end_for(i)

Figure 2.6: Pseudo code of Boostexter's model building

```
for l=0; to |C|
    f_l = 0;
    for t=0 to T
        if (h_word[t] ∈ d) ⇒ f_l += c1[t][l];
        if (h_word[t] ∉ d) ⇒ f_l += c0[t][l];
    end_for(t)
end_for(l)
```

Figure 2.7: Pseudo code for classifying a new document using a given Boostexter model

Chapter 3
Intelligent support

This chapter deals with methods for reducing annotation cost by *sample selection*. The hope is that, by using and combining recent machine learning methods, one can develop more efficient text classifiers. It is expensive to acquire a sufficient number of labeled examples for training. In this approach, during the training the learning program examines many unlabeled examples and selects for labeling only those that are more informative at each stage. This avoids redundantly labeling examples that contribute little new information. We demonstrate that the accuracy of learned text classifiers can be improved by augmenting a small number of labeled training documents with a large pool of unlabeled documents. This chapter also presents methods to use the user's prior knowledge during the training process. Finally, we introduce how to grant feedbacks to users. These user feedbacks are rich, containing those words which have been significant in the decision making.

3.1 Initial sample selection

Using any kind of algorithm, we need to have a set of labeled examples in hand before the classification process begins. More precisely, we need to have at least one example for each target class. This section describes how to choose these "initial" documents.

3.1.1 Random selection

We simply choose a given number of documents randomly. Obviously, this is the simplest - and most common - way of selecting examples. Labeling a document is a costly operation, therefore a "smart" pre-selection or heuristic could improve the performance efficiently. The following section describes a possible solution.

3.1.2 Selection using k-means clustering

K-means clustering is an algorithm for partitioning (or clustering) N data points into K disjoint subsets. The algorithm consist of a simple re-estimation procedure as follows. First, the data points are assigned at random to the K sets. Then the centroid is computed for each set, and each document is moved to the cluster closest to it. These step is alternated until a stopping criterion is met, i.e., when there is no further change in the assignment of the data points.

K-means algorithm properties:

- the clusters are non-hierarchical and they do not overlap
- every member of a cluster is closer to its cluster than any other clusters

To complete this clustering, we need to measure the distance between two documents. Note that as it is described in 2.3.1, we have a vocabulary of unique words $\{w_i\}$. For each document α we construct a feature vector v^α, where the components i are determined by the frequency of which word w_i occurs in the document. Following standard practice [20] we use the $TFIDF$ ($term\ frequency \times inverse\ document\ frequency$) weighting scheme:

$$v_i = F_t(w_i, \alpha) \log(\frac{N}{F_d(w_i)}) \tag{3.1}$$

where $F_t(w_i, \alpha)$ denotes the number of times the word w_i occurs in document d_α. N is the number of documents in the training set and $F_d(w_i)$ the document frequency of w_i, counting the number of documents where w_i occurs.
The similarity between document α and β is computed with the cosine metric:

$$d_{\alpha\beta} = \cos(v^\alpha, v^\beta) = v'^\alpha v'^\beta \tag{3.2}$$

where

$$v' = \frac{v}{\|v\|} \tag{3.3}$$

Put into practice, an efficient way is to store the normalized document feature vectors (v'). Figure 3.1 contains the scheme of the k-means clustering algorithm with the above described weighting and metric.

Experiments show that k-means is a good initialization method, therefore we add a faster implementation. Figure 3.2 shows the scheme of a modified k-means clustering method, which is $O(\log n)$ in time, instead of $O(n)$.
K-means clustering is a very expensive algorithm both in time and in memory usage. The size of the normalized document feature vector is equal to the number of distinct words in the vocabulary. Known that the cost of calculating the distance between two documents is also growing proportionally to the size of the vocabulary.
As it seems in Chapter 5 k-means clustering evidently surpasses the random selection method in accuracy.

3.2 Intelligent sample selection

In this section we describe a different approach for sample selection, usually called *active learning* in the literature. Most text mining methods are based on a fixed number of training examples, usually chosen randomly from the set of documents. In this approach, during the training the learning program examines many unlabeled

Given: $\{w_i\}$ vocabulary
$\forall \alpha \in \mathcal{D} : \vec{F_t}(\alpha) = (F_t(w_1, \alpha), F_t(w_2, \alpha), ...)$ term frequency vector

1. Calculating normalized document feature vectors
   $\forall \alpha \in \mathcal{D}$:
   (a) v^α document feature vector, where
   $$v_i^\alpha = F_t(w_i, \alpha) \log(\frac{N}{F_d(w_i)})$$
   (b) $v^{'\alpha}$ normalized document feature vector, where
   $$v^{'\alpha} = \frac{v^\alpha}{\|v^\alpha\|}$$
2. k-means clustering
   (a) The documents are partitioned into K clusters $(P_1, ...P_k)$ randomly (clusters have roughly the same number of points)
   (b) calculate the center of each cluster
   $$\forall i \in \{1..k\} : \vec{v_{P_i}} = \frac{\sum_{\alpha \in P_i} v^{'\alpha}}{|P_i|}$$
   (c) for each document α: calculate the distance from the center of each cluster
   $$\forall i \in \{1..k\} : d_{\alpha P_i} = 1 - v^{'\alpha} v_{P_i}$$
   If it is closest to its own cluster, leave it where it is. If the document is not closest to its own cluster, move it into the closest cluster.
   (d) Repeat (b)-(c) while centroids do not change
   (e) Take the document from each cluster closest to the center as an initial example to be labeled

Figure 3.1: Example selection using k-means clustering and TFIDF weighting with cosine metric

1. Let $\mathbb{P} = \{\mathcal{D}\}$
2. for $i := 1$ to $k - 1$ do
   (a) Select $P \in \mathbb{P}$ with maximal cardinality
   (b) Choose randomly two data points in P as starting centroids $v\vec{}_{P_1}$ and $v\vec{}_{P_2}$
   (c) assign each point in P to the closest centroid, splitting thus P in two clusters P_1 and P_2
   (d) (Re-)calculate the cluster centroids $v\vec{}_{P_1}$ and $v\vec{}_{P_2}$ of P_1 and P_2
   (e) Repeat (c)-(d) until the centroids do not change anymore
   (f) Let $\mathbb{P} := (P \backslash \{P\}) \cup \{P_1, P_2\}$

Figure 3.2: A faster implementation of k-means clustering

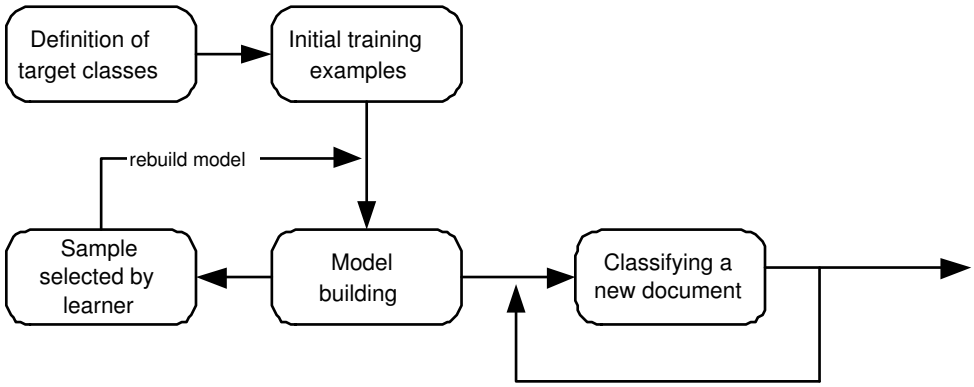

Figure 3.3: Text mining process expanded with active learning

examples and selects for labeling only those that are most informative at each stage. This avoids redundantly labeling examples that add little new information. Figure 3.3 shows how active learning expands the text mining process.
Note that we cannot avoid to have a number of initial examples in hand, at least one for each target class, to build a classification model.
We describe a method to produce a ranking between documents according to classification's result. Then 3 different active learning algorithms and approaches are presented.

3.2.1 Ranking

Let us define a ranking between documents. Documents are sorted according to the score values produced by the classifier. This sorting is created for each target class, then a rank vector $r = [r_1, ...r_{|C|}]$ is created for each document. The ith element of

the rank vector, r_i, shows that there are $r_i - 1$ documents exist with higher score for class c_i. Accordingly, for a given target class c_i, the most confidently classified document has $r_i = 0$, while document with the lowest score is getting $r_i = |d \in \mathcal{D}|$. Rank values are re-scaled between $0..1 \in \mathbb{R}$, divided each component by the number of all documents:

$$r' = [\frac{r_1}{|\{d \in \mathcal{D}\}|}, ... \frac{r_{|C|}}{|\{d \in \mathcal{D}\}|}]$$

3.2.2 Combined models

This approach is based on the idea of combining different classification algorithms. Given a set of training examples a model is built by each classifier, then those documents are selected to be labeled, on which they disagree most! In our case it means a NaiveBayes and a Boostexter model. (Building two models of the same kind is meaningless since they are equivalent for the same training set).
Let $r_i'^1$ is the rank of $d \in \mathcal{D}, i \in \{1..|C|\}$ produced by classifier (1) and $r_i'^2$ is the rank produced by classifier (2).

$$m = \sum_{i=1..|C|} \frac{(r_i'^1 - r_i'^2)^2}{2} \tag{3.4}$$

The score of the voting is defined in (3.4) where $m_1 > m_2$ means that classifier (1) and (2) disagree on m_1 more than on m_2. Documents with the higher m value are selected to be labeled in each active learning iteration.

This approach can be generalized for n classifiers. In this case we compare the ranks to the average rank produced by n classifiers.

$$m = \sum_{i=1..|C|} \sum_{j=1..n} (R_i - r_i'^j)^2 \tag{3.5}$$

$$R_i = \frac{\sum_{j=1..n} r_i'^j}{n} \ \forall i \in \{1..|C|\} \tag{3.6}$$

Note that in the $n = 2$ case

$$\begin{aligned} m &= \sum_{i=1..|C|} (R_i - r_i'^1)^2 + (R_i - r_i'^2)^2 \\ &= \sum_{i=1..|C|} (R_i)^2 - 2R_i r_i'^1 + (r_i'^1)^2 + (R_i)^2 - 2R_i r_i'^2 + (r_i'^2)^2 \\ &= \sum_{i=1..|C|} 2(R_i)^2 - 2R_i(r_i'^1 + r_i'^2) + (r_i'^1)^2 + (r_i'^2)^2 \qquad (3.7) \\ R_i &= \frac{r_i'^1 + r_i'^2}{2} \ \forall i \in \{1..|C|\} \qquad (3.8) \\ & \qquad (3.9) \end{aligned}$$

Using (3.8) to rewrite (3.7):

$$\begin{aligned} m &= \sum_{i=1..|C|} 2(\frac{r_i'^1 + r_i'^2}{2})^2 - 2\frac{r_i'^1 + r_i'^2}{2}(r_i'^1 + r_i'^2) + (r_i'^1)^2 + (r_i'^2)^2 \\ &= \sum_{i=1..|C|} \frac{(r_i'^1 + r_i'^2)^2}{2} - (r_i'^1 + r_i'^2)^2 + (r_i'^1)^2 + (r_i'^2)^2 \\ &= \sum_{i=1..|C|} \frac{(r_i'^1 + r_i'^2)^2}{2} - ((r_i'^1)^2 + 2r_i'^1 r_i'^2 + (r_i'^2)^2) + (r_i'^1)^2 + (r_i'^2)^2 \\ &= \sum_{i=1..|C|} \frac{(r_i'^1 + r_i'^2)^2 - 4r_i'^1 r_i'^2}{2} = \sum_{i=1..|C|} \frac{(r_i'^1)^2 - 2r_i'^1 r_i'^2 + (r_i'^2)^2}{2} \\ &= \sum_{i=1..|C|} \frac{(r_i'^1 - r_i'^2)^2}{2} \end{aligned} \tag{3.10}$$

which is equal to (3.4)

3.2.3 Lowest variance

This section describes a possible method for measuring the variance of the ranks. A variance is defined in the following way:

$$v = \sum_{i=1..|C|} (V - r_i')^2 \tag{3.11}$$

$$V = \frac{\sum_{i=1..|C|} r_i'}{|C|} \tag{3.12}$$

where r_i' is the rank of $d \in \mathcal{D}, i \in \{1..|C|\}$ produced by the classifier.
The idea behind the formula is to label those documents which produced almost the same rank values for each target class. According to (3.11) a lower v value is produced for these ones.
In spice of the simplicity of this suggestion it produced very convincing results. See Section 5.3.3 for details.

3.2.4 Group membership factor

The use of the ranks makes us possible to measure the confidence that a document is a member of its predicted target class. Let be $d_c \in \{1..|C|\} \quad d_c = \min(r_1', ..., r_{|C|}')$ the target class with the lowest r_i' value, where r_i' is the rank of $d \in \mathcal{D}, i \in \{1..|C|\}$ produced by the classifier. A confidence that the document belongs to this class is denoted by b while the confidence of being not a member of other classes is denoted by $\overline{b}$ and calculated in the following way:

$$b = r_{d_c}' \tag{3.13}$$

$$\overline{b} = \sum_{i=1..|C|, i \neq d_c} (1 - r_i') \tag{3.14}$$

r'_1	r'_2	r'_3	$\bar{r}_1$	$\bar{r}_2$	$\bar{r}_3$	CM	LW	GMF
0.10	0.60	0.50	0.20	0.70	0.45	**0.011**	0.140	1.000
0.30	0.40	0.45	0.35	0.40	0.50	0.003	**0.012**	1.450
0.60	**0.34**	0.40	0.60	0.34	0.40	0.000	0.037	1.340
0.90	0.60	**0.53**	0.95	0.50	0.62	0.010	0.077	1.030
0.24	**0.21**	0.78	0.20	0.25	0.80	0.002	0.206	1.190
0.21	**0.15**	0.47	0.23	0.10	0.60	0.010	0.058	**1.470**

Table 3.1: Comparison of different active learning methods. The table contains an example set of classified documents. $r'_1..r'_3$ denotes the rank values produced by the classifier. $\bar{r}_1..\bar{r}_3$ are produced by the 2nd classifier. We pay respect to them only in case of CM. CM denotes combined models, LW denotes lowest variance while GMF means group membership factor. In the first three columns the lowest rank values are highlighted. In CM, LW and GMF columns the element chosen by the active learning method is highlighted.

The group membership factor is the linear combination of the two confidences:

$$f = b + \bar{b} \tag{3.15}$$

In this interpretation the lower f value means that a document is classified unequivocally.

3.3 Use of unlabeled data

Most computational models of supervised learning rely on labeled training examples and ignore the possible role of unlabeled data. However, as it is shown in [14], [17] unlabeled data can significantly improve learning accuracy.

How can we use unlabeled data to boost learning accuracy? The classifier is allowed to examine the unlabeled data and pick its most confidently predicted examples, and add these to the set of labeled examples. The process is repeated as long as desired.

The issue we have to deal with is how to obtain the confidence of a given document's classification. We describe two ways of it. An obvious solution is to sort documents according to the score attached by the classifier. Documents with highest score are selected and added to the labeled examples. This method is denoted with HS in the following sections.

A more complex way is to use the ranks and calculate the variance values as it is described in (3.11), Section 3.2.3. The difference is that while in active learning we are searching for the difficultly classified documents, now we need the opposite. In this case the higher variance is better. A higher variance value means that there is a significant class exists. This method is denoted with HV.

There is an other issue we need to consider, namely the size of the documents' set, used as unlabeled data. The answer is given by experiments. As we shall see in Section 5.3.5, this method can indeed dramatically improve the accuracy of

category	words
electronics	connector, electric, electrical, power, voltage, transformer, device,...
politics	constitution, president, government, taxpayer, ...
religion	God, Christ, Jesus, Bible, faith, angel, heaven, hell ...

Table 3.2: Example of user-defined prior knowledge

a classifier, especially when there are only a few labeled documents. But it may happen that the incorporation of unlabeled data decreases, rather than increases, classification accuracy. Thus it is necessary to choose the size of the unlabeled data set carefully.

3.4 Prior user knowledge

Sample document's class labels are determined by a user - independently of being an initial example or chosen by an active learning method. It pretends some 'background' knowledge about the given data set. This leads to the suggestion that this prior user knowledge can be used to improve the classification's accuracy. The question is how to represent this information. A practicable way is described in this section.
The user is asked to enumerate words for each target which are predicted to be frequent and denotes the category with high probability. Evidently it is meaningless to list words which identify the class in all cases but occur just in a few documents. Table 3.2 contains a possible user-defined knowledge in a form of enumerating words. Apparently, there can be common elements (for example the category 'religion' and 'atheism' motivate almost the same words), but it is enough to enumerate these words only in one of the relevant targets (we shall see later why).

Hereunder we show that this knowledge can be used to improve classification's accuracy. We use the assumption that the enumerated words are frequent ones and relevant only for a few targets (mostly one). In this case it is reasonable to weight them since they hold relevant information about the target class.
It proposes two questions. (1) what should be the size of the attached weight and (2) is it possible to simulate this background knowledge? Answering (1) is quite simple, since we choose it to be constant and experiments give the concrete answer. We describe a method in the following which gives a possible solution for (2).

Given a set of documents we count the frequency of each word for each target. w_i^j denotes the frequency of the ith word of the vocabulary in target j ($j \in \{1..|C|\}$). w_i is the sum frequency of the word: $w_i = \sum_{j=1..|C|} w_i^j$. Words are sorted according to w_i and a rank value is attached to each one. r_i is the rank of the ith word. Ranks are scaled between 0 and 1, the most frequent word's rank is 1, while the less frequent's rank is 0.

score	word	w_i	w_i^{1-10}	w_i^{11}	w_i^{12}	w_i^{13}	w_i^{14}	w_i^{15}	w_i^{16}	w_i^{17}	w_i^{18}	...
1.963	clipper	5674	...	0	5500	0	0	34	0	108	0	...
1.957	israeli	2312	...	0	0	0	0	0	4	1	2300	...
1.956	nhl	2084	...	2083	0	0	0	0	0	0	0	...
1.940	tapped	1789	...	0	1782	2	0	0	1	0	2	...
1.938	hockey	1967	...	1942	0	0	0	0	0	0	2	...

Table 3.3: Example of calculated word scores for simulating prior user knowledge. w_i^j denotes the frequencies for targets while w_i is the sum of them.

A score is calculated for each word (w_i) in the following way:

$$s_i = r_i + \frac{\max_{j=1..|C|} w_i^j}{w_i} \tag{3.16}$$

Words with the highest scores are selected as prior user words. In practice we calculate the s_i values for only the top 30% frequent words. Table 3.3 shows an example for the calculated scores.

3.5 User feedback

`iTM` is designed not only to help the model building process but to grant feedbacks to users. In this section we summarize which features can be used for providing this. First of all, we have to give information about the representation of the classified documents. A document can be considered as a sequence of words, tokens and punctuation. Whereas we use only a set of words among them (elements of the vocabulary) we should indicate these ones. Moreover, there are weighted words (e.g. subject of the e-mail messages, html tags, etc) which also should be discriminated. Probably the most relevant information is held by the significant words of the model. In case of Naive Bayes these come with the highest class conditional probabilities. In case of Boostexter these are the words chosen as weak classifiers. These words should be denoted, moreover the target class, where they belong also should be indicated.
Finally, it could be also helpful to mark the words, defined by the user as prior knowledge.
Section 4.7 describes the visualization of these features.

Chapter 4
System overview

Figure 4.1 shows the global architecture of the `iTM` system. This chapter presents the description of the different components.

4.1 Graphical Interface

The graphical interface is a frame which holds the components together. All functions for users can be reached through the GUI.

4.2 Source manager

`iTM` can handle several kind of textual documents (text files, e-mails, html pages). This component is responsible for the definition of data sets. Note that different kind of data can be used at the same time (for example some newsgroups or e-mail messages and a directory with html pages).

4.3 Vocabulary manager

This component is responsible for managing the vocabulary. It displays the words of the vocabulary together with the corresponding frequencies. Words below or above a given frequency can be neglected here. We can also limit the vocabulary in a given number of words.

4.4 Classification

The parameters of the text classification can be set here. These are the following:

Classifier:
We can choose between the implemented classifiers. As it is described in Section 2.4, these are Naive Bayes and Boostexter. [1]

Mode:
Two possibilities are given here: interactive and simulation. Interactive is used to classify a set of documents of which the target class labels are

[1] `naivebayes.conf` and `boostexter.conf` contain additional parameters of the classifiers. These can be tuned without re-building the source code of `iTM`.

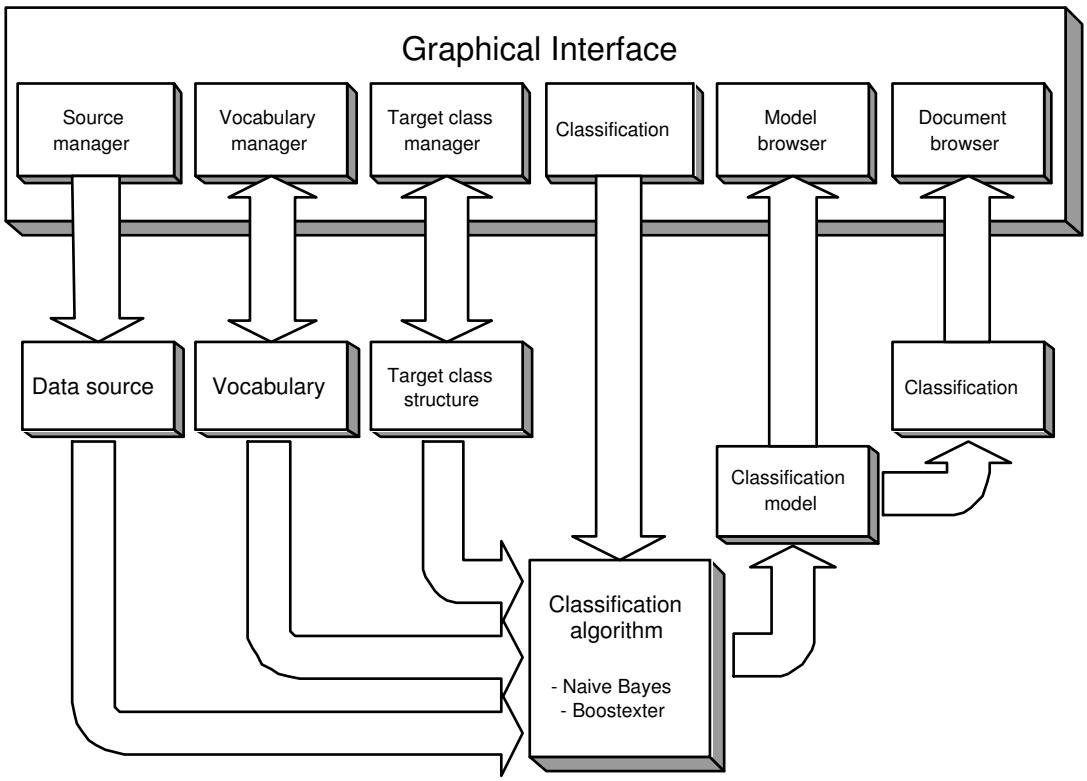

Figure 4.1: Global architecture of the `iTM` system

unknown. This is the normal, "real-world" use of the system. Simulation mode makes us possible to measure the accuracy of the classifier with the selected settings. In this mode a class label has to attached to each data source. Then a program "simulates" the user's selection and compares the results to the real class labels. There is an other possibility for running experiments with more options. It is called *batch mode* and described in Section 4.8.

Initial sample selection:
The number of initial sample documents and the mode of selection are defined in this option. As it is shown in Section 3.1, we need to have a number of initial examples before the classification process begins. `iTM` provides two ways for the selection of these samples: random and using k-means clustering.

Intelligent sample selection:
Use of this option makes us possible to choose an active learning method and set the parameters of intelligent sample selection. A given number of documents will be asked for labeling as many times as we defined in the iterations field. The different intelligent methods are described in Section 3.2.

Use unlabeled data:
Unlabeled data can be used to improve classification's performance. Size of the used unlabeled documents' set is defined here.

Prior user knowledge:
Use of this option provides a surface for users to define background knowledge in the form of enumerating words.

Continue classification:
A previously saved classification can be loaded and/or continued.

4.5 Target class manager

Algorithms, described in Chapter 2.4, are designed to classify documents into a fixed list of possible categories. In our approach categories can be defined not only as an enumeration but also in a tree structure. Moreover `iTM` allows users to change the structure of this target tree real-time. Note that simple enumeration of the target classes corresponds to a 1-height tree, where each node is connected to the root.
This component is responsible for the maintenance of the target classes structure. Empty categories are neglected before each active learning iteration, then restored. A visual tool helps the user to add/remove nodes of the tree.

4.6 Model browser

This component is responsible for displaying the classification model built by the `iTM` system. In case of Naive Bayes classifier the target probability variables ($P(c_j)$) and the class conditional probabilities ($P(w_k|c_j)$) are listed. In case of Boostexter the real-valued predictions (c_{0l}, c_{1l}) are displayed. The model -together with the documents' classification- can be saved to and loaded from file.

4.7 Document browser

The classified documents can be browsed through this tool. The followings are displayed for each:

Classification:
The document's classification. It shows the predicted target class and the rank/score values for all targets.

Content:
The document's content, displayed according to its type. In case of flat text or email messages it is equivalent to the source. If the document's type is html then the page is displayed graphically, as in a browser.

Source:
The source of the document.

Words list:
The internal representation of the document. Since we use bag-of-words representation it is a list of words together with word frequencies.

Feedback:
A graphical feedback for the user. We use different colors and notation to report which features were taken into account during the classification process.

4.8 Batch mode

In many cases we are interested in the result of not just one but more runs. Besides it may happen that we would like to run experiments with the same configuration but with different size of sample documents. To follow the accuracy of an active learning method also could be beneficial. Batch mode is designed to run experiments requested times with a given configuration. A loop can be defined for the size of the training set. The program records the accuracy after each iteration. It also works with active learning. In the end a `matlab` file is created with the detailed results. Results and plots found in Chapter 5 are all created with this batch tool. Note that it corresponds to the simulation mode of the GUI but makes possible several runs

and variable training set size. Section 7.1 in the Appendix shows an example batch configuration file.

Chapter 5
Experiments and results

In this chapter, we describe and analyze the experiments we performed using the *intelligent support* methods that were described in Chapter 3. We intend to illustrate that the number of labeled documents can be decreased significantly with the use of these ideas. We also describe a study about classifying a web domain with our `iTM` system.

5.1 Data set

20 newsgroups
The 20 Newsgroups data set is a collection of approximately 20,000 newsgroup documents, partitioned (nearly) evenly across 20 different newsgroups. It was originally collected by Ken Lang. The 20 newsgroups collection has become a popular data set for experiments in text applications of machine learning techniques, such as text classification and text clustering. Many of the categories fall into confusable clusters; for example, five of them are `comp.*` discussion groups, and three of them discuss religion. The corpus is publicly available from the web page: `http://people.csail.mit.edu/u/j/jrennie/public_html/20Newsgroups/`.
We used a cleaned-up version of this dataset ("18828"), which does not include cross-posts (duplicates) and includes only the "From" and "Subject" headers.

5.2 Implementation details

The results are created with the `iTM` system's batch mode component. (Batch mode is described in Section 4.8). Some of our algorithms are nondeterministic and require random selections. Because by chance a random selection may be very lucky or not we repeated every experiment that involves random selection 10 times and took the average. The experiments were performed on an intel Pentium 4 PC with 512MB RAM running at 1.6Ghz.

5.3 Results

This section presents end explains the results of different configurations and active learning methods, used for the classification of the 20-newsgroups dataset. Since

configuration	lower bound	upper bound	sum words
Vocabulary 1	15	15000	17215
Vocabulary 2	20	10000	14149
Vocabulary 3	50	10000	7362
Vocabulary 4	100	8000	4304

Table 5.1: Different vocabulary configurations for classifying the 20-newsgroups data set

there are several constants and parameters which can be tuned, we hold down to detail only the most significant ones. Full particulars can be found in the corresponding chapters of the Appendix.

5.3.1 Size of the vocabulary

The first question we have to deal with is the size of the vocabulary. As it is described in Section 2.3 words with very low and very high frequency are neglected. Putting it into practice simply means that we need two boundaries for word frequencies. Table 5.1 contains four different configurations. As it is shown in Figure 5.1 there are well-marked differences in the Naive Bayes classifier's behavior in the four cases. Vocabulary with a greater size leads to higher accuracy while the precision is lower in the first section of the plot. In to contradistinction, vocabulary with a smaller size achieves higher accuracy in the beginning but performs worse as the number of training examples grows. Note that these examples are selected randomly.
Since we are to reduce the number of training examples, in the next experiments we label not more than 15% of the documents, which means 2900 samples maximally. Accordingly, we use the 4th vocabulary configuration. Besides its better performance in the desired segment of the plot it has also better performance in time. Smaller vocabulary effects that less computation is needed.
Actually, we managed to increase the performance of a classification algorithm by simply reducing the size of the vocabulary but we do not take it as an improvement. We consider it as base settings and the starting point of our experiments. Random selection will denote the accuracy achieved by vocabulary configuration 4 in the following.

5.3.2 Representation

Whereas this data set contains only e-mail documents, the only issue we have to deal with is the weight attached to headers. The header contains a 'subject' and a 'from' field. The 'from' field contains only the senders name or email address (or both), thus no information about the content, therefore we ignore it. Weights are attached only to the words of the 'subject' field.
Figure 5.2 shows the accuracy achieved by the Naive Bayes classifier, using different weights on the subject field of the newsgroup messages. It clearly proves that the subject field contains relevant information about the content. Neglecting it (attach-

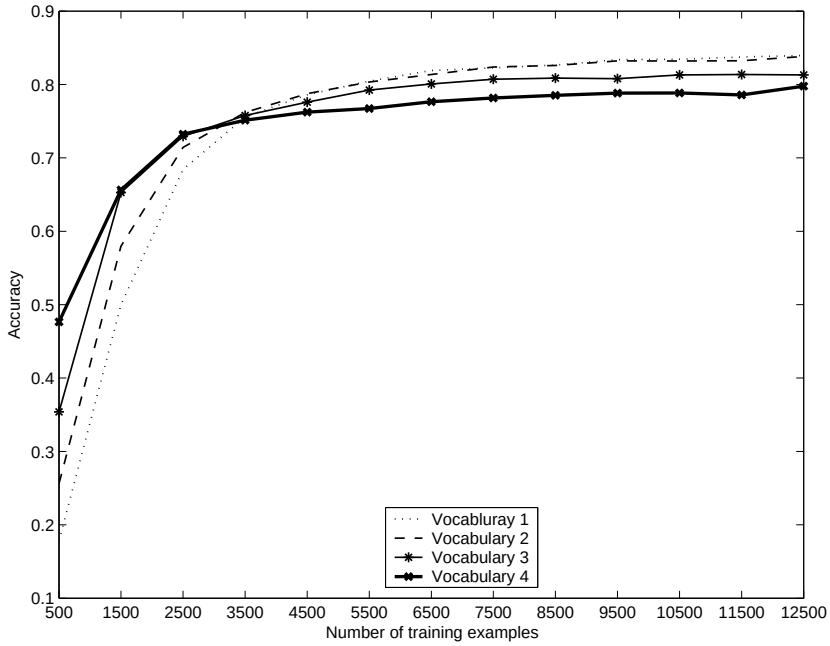

Figure 5.1: Accuracy of Naive Bayes classifier using different vocabulary configurations and random example selection

ing 0 weight) performs much worse than any other setting. Attaching weights to the subject field can improve classification accuracy up to 8%.
Based on these observations, we use weight 10 in the following. More detailed results can be found in the Appendix, in Table 7.2.

5.3.3 Intelligent sample selection

This approach is based on the following idea. A classification model is built from the initially selected random examples. Then the learning program examines many unlabeled documents and selects for labeling only those that are most informative at each stage. This step is repeated several times.
In practice it means that firstly 100 examples are selected randomly. Then a classification model is built and the active learner selects a number of examples to be labeled. We have tried two configurations. To add 8 times 350 examples or 16 times 175 examples. The second configuration performed better in all cases. It corresponds to our expectations since the model is getting more adequate after each iteration. At the same time the number of iterations can not be increased boundlessly because model building is a time consuming process.
Figure 5.3 shows the results of different active learning methods. We used the same, previously described settings (vocabulary size, representation) for all cases, including random selection.
GMF performed better from the three methods. With 3000 labeled examples it reaches 80% accuracy while random selection achieves 76.6%. This represents a 18% reduction in classification error. Note that the benefits of these active learning

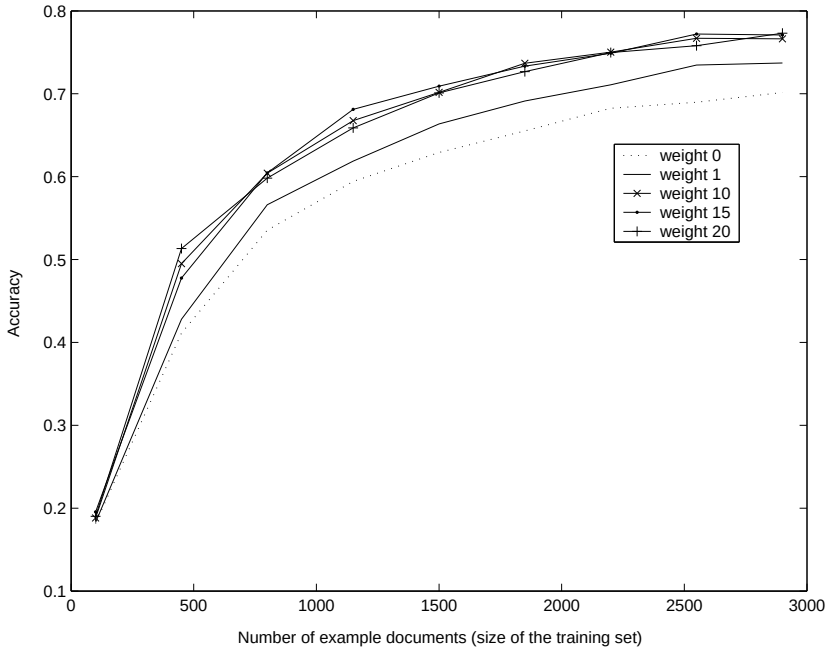

Figure 5.2: Accuracy of the Naive Bayes classifier using different weights on the subject field of the newsgroup messages. Examples are selected randomly.

methods can be fully realized after 3000 labeled examples.
Naive Bayes with normal sized vocabulary (voc1 in Section 5.3.1) achieves 80% accuracy with 5500 labeled exmaples. GMF reaches the same accuracy with only 3000 labeled examples. It means that the number of manually labeled examples can be decreased by 45%.
More detailed results can be found in the Appendix, in Table 7.3.

5.3.4 Initial sample selection

Using k-means clustering instead of random initial example selection is a right alternative. At the same time we have to take into account that it is a very expensive calculation which even grows with the number of examples. Therefore we use it together with active learning. In that case there is an initialization at the beginning when the first set of training examples are selected with this clustering method. Then the active learning method is responsible for selecting the following samples.
Figure 5.4 shows the comparison of random selection and k-means clustering. With 100 labeled documents random selection achieves 18.8% while k-means clustering reaches more than 10% higher. Table 7.4 contains more details. These results demonstrate that k-means clustering improve classification accuracy and reduce the need for labeled training examples.

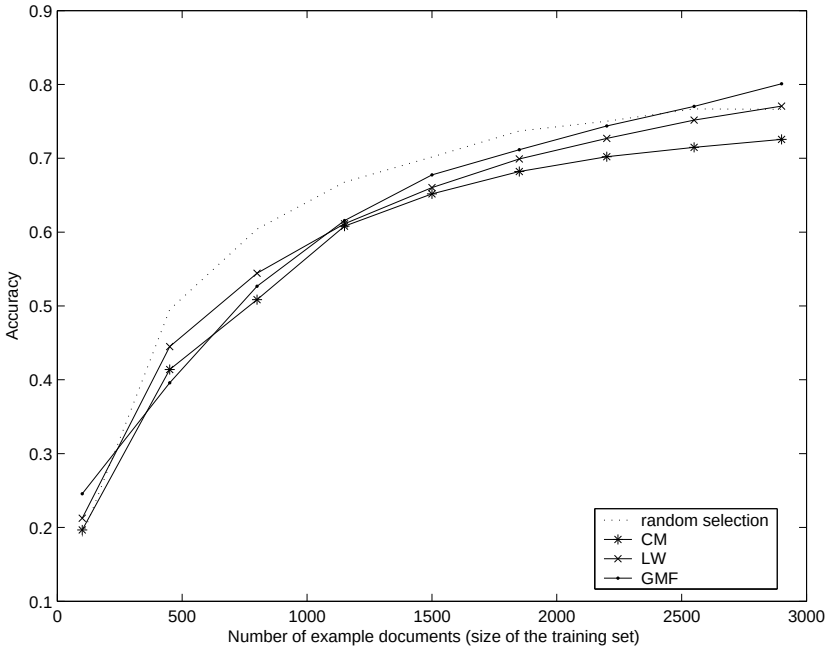

Figure 5.3: Comparison of different active learning methods.

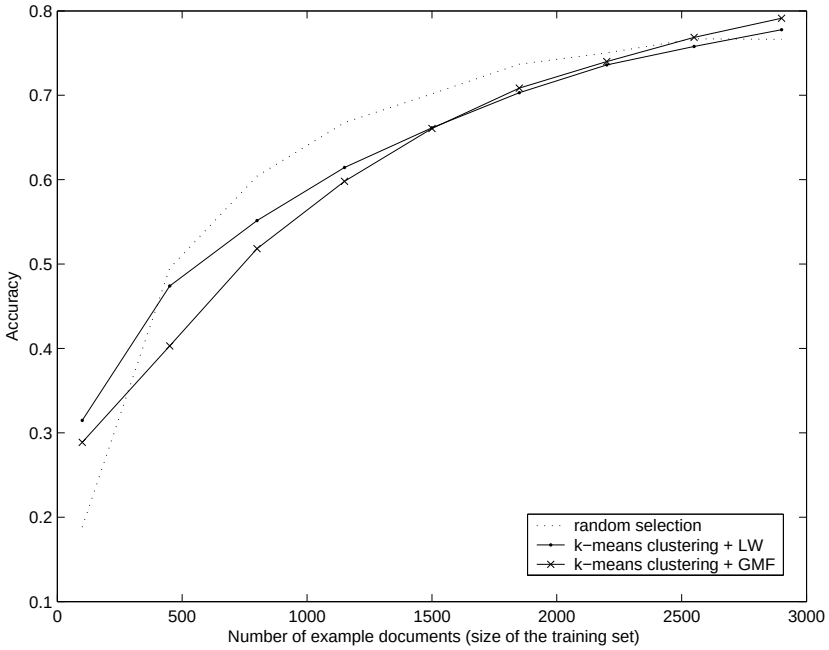

Figure 5.4: Comparison of k-means clustering and random selection. K-means clustering is used for selecting only the first 100 examples, additional samples are selected by an active learning method.

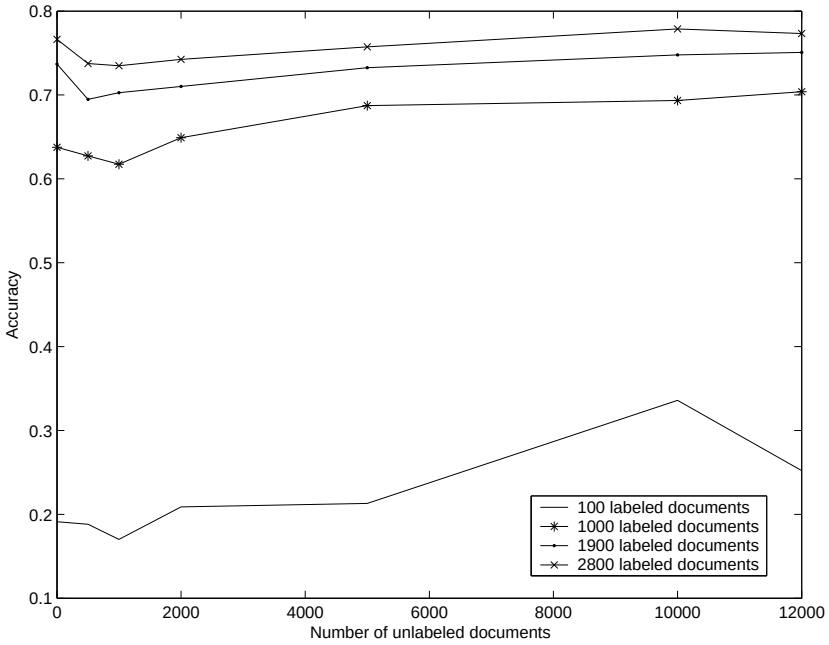

Figure 5.5: Accuracy of the Naive Bayes classifier while varying the number of unlabeled documents. The effect is shown with 4 different amounts of labeled documents. Unlabeled documents were selected by using the HV (highest variance) method.

5.3.5 Use of unlabeled data

In Figure 5.5 we consider the effect of varying the amount of unlabeled data using the Naive Bayes classifier. For four different quantities of labeled documents, we hold the number of labeled documents constant, and vary the number of unlabeled documents in the horizontal axis. Naturally, having more unlabeled data helps, and it helps more when there is less labeled data. Notice that adding a small amount of unlabeled data actually hurts performance. [17] hypothesize that this is caused by the Naive Bayes independence assumption.
Section 3.3 describes two methods (HS,HV) for selecting unlabeled documents. Figure 5.6 shows the effect of using these methods.
The results show that the use of unlabeled data improves classification accuracy. When there are 800 labeled documents traditional Naive Bayes with random sample selection attains 60.4% accuracy, while HS reaches 67.6%. When there is a lot of labeled data having unlabeled data does not help nearly as much because there is enough labeled data to accurately estimate class labels. There is no significant difference in the behavior of the two selection methods. Table 7.5 and 7.6 contain more detailed results.

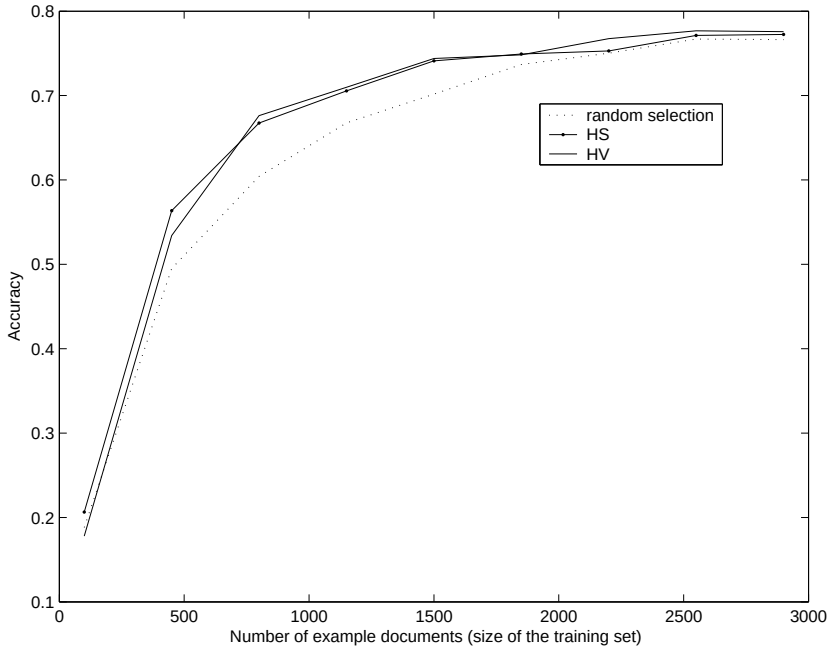

Figure 5.6: Comparison of random selection and use of unlabeled data. The size of the unlabeled data set is 10000 in both cases.

5.3.6 Prior user knowledge

We have simulated the prior user knowledge in such a manner that described in Section 3.4. It means that a score is calculated for each word of the vocabulary. Top 381 words with highest score were selected as prior user knowledge. Then a weight is attached to these words.

Figure 5.7 shows the accuracy of the Naive Bayes classifier with different weights attached to user words. Note that the e-mail headers are also weighted. It is shown that the discrimination of prior user words improves classification accuracy, especially when there are only a few labeled documents. When there are 100 labeled documents, random selection attains 18.8% accuracy, while attaching weight 10 reaches 31.2%. This represents more than 12% improvement in classification accuracy. Notice that attaching too heavy weights actually hurts performance when there is a lot of labeled data.

Section 7.2.1 contains detailed information about selected words and results.

5.3.7 Summary

Our primary goal is to achieve high accuracy with labeling only a few examples. Therefore we combine all previously described features, namely: k-means clustering for initialization, intelligent sample selection using GMF method, use of unlabeled data, and prior user knowledge. Figure 5.8 shows the accuracy of this configuration compared to random selection. With 100 labeled examples random selection achieves 18.8% accuracy, while using all features reaches 42%. It represents 23% improvement

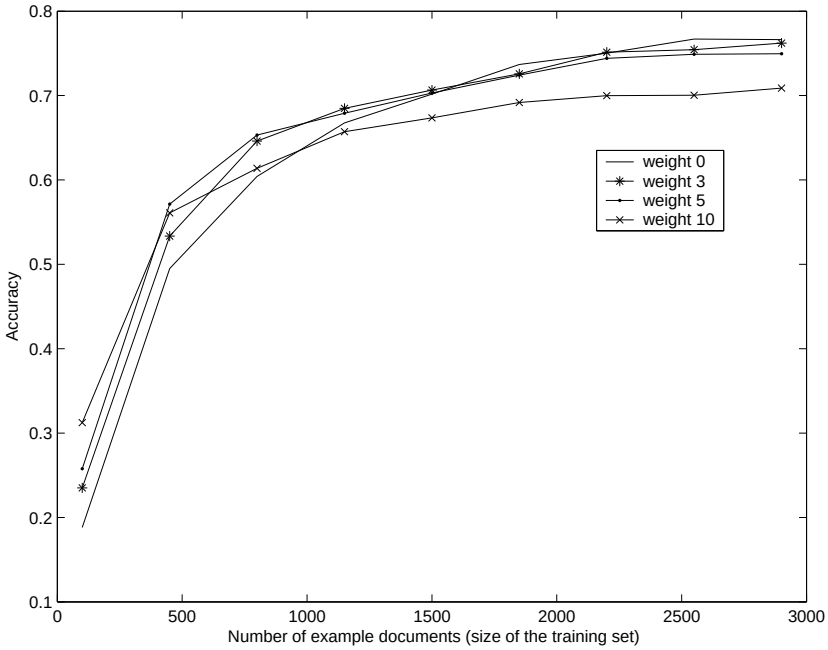

Figure 5.7: Accuracy of the Naive Bayes classifier using different different weights on prior user knowledge. Example documents are selected randomly.

in classification accuracy. Note that random selection uses the idea of reduced sized vocabulary (Section 5.3.1). Comparing to standard Naive Bayes the improvement is even convincing. With 500 labeled examples standard Naive Bayes achieves 17.8% accuracy while our methods reaches more than three times higher.
Table 7.8 contains detailed results.

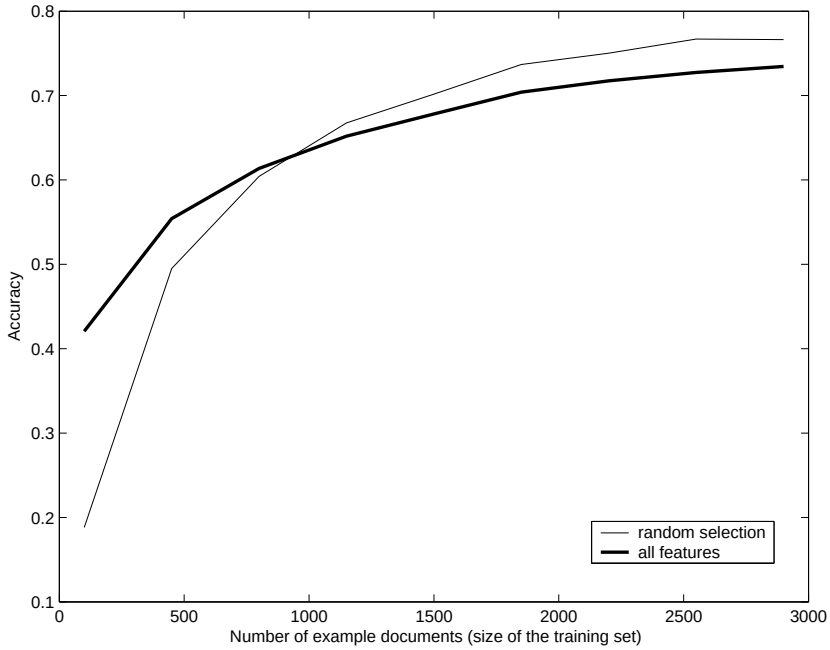

Figure 5.8: Comparison of random selection and the use of all `iTM` features. 'All features' includes k-means clustering for initialization, intelligent sample selection using GMF, use of unlabeled data (10000 documents), and prior user knowledge.

5.4 Classifying www.cs.vu.nl

This experiment is to illustrate the use of the `iTM` system on a real-word data set and also to produce data for further data mining tasks ([9]). The data set is created from the html pages of the Vrije Universiteit, Amsterdam Computer Science faculty (`http://www.cs.vu.nl`). The corpus contains all publicly available `htm` and `html` files from the given domain.

5.4.1 Configuration

The data set consists of 13011 html documents from the given domain and holds 157628 different words. Words with very low (< 100) and very high (> 5000) frequency were removed. The vocabulary finally contained 5417 different elements. The experiment was performed on an intel Pentium 4 PC with 512MB RAM running at 1.6Ghz. In spite of the relatively large amount of data, the hardware requirements are not surpass a home PC's configuration.

5.4.2 Categorization

Our goal is attach class labels to each document based on its content. Figure 5.9 shows the architecture of the possible targets. The description of the classes is the following:

activity:

Can be science conference, exhibition, concert, trip for international students or any other happening which has connection to the university. Youth organization (e.g. STORM) pages also belong here.

course:
Course pages, usually containing the course's description, lecture slides, recommended readings and assignments.

department:
Pages of the university's departments.

empty:
Pages that hold absolutely no information about the content. Usually framesets, blank, moved or redirected pages. It can also contain images, but without any textual reference.

person/faculty:
Faculty member's pages. Usually very formal, refering to fields of work, to the own department and to research projects.

person/faculty/publications:
A list of selected papers or publications (containing at least the 'abstract' section).

person/student:
Student's web pages. Usually containing personal information (hobby, song's lyrics, etc), sometimes links to friends or course pages.

photo:
Pages containing no textual information for identification, just one or more pictures. Can be a menu part of a frameset, a personal photo album, lecture slide or an index page containing an image with 'under construction' or 'this page has moved' message.

project:
Usually means a research project of the university or a department.

reference:
Reference or manual page for an operation system, programming language, library or internal project. It also can be a course slide, source code or a bibtex item.

other language:
Pages written in different (non English or Dutch) language.

Since the language of the pages is English or Dutch, it provides us an other aspect for classification. We expand the previous structure by separating the corresponding labels into English and Dutch (there were no Dutch pages for some categories and there are labels with no language context, like empty and photo). Figure 5.10 contains the expanded structure.

5.4.3 Classification

Documents are represented as HTML pages in such a manner that described in Section 2.3.2. We applied the default settings of `iTM` for the tags weighting. We used the Naive Bayes algorithm with intelligent sample selection for classification.

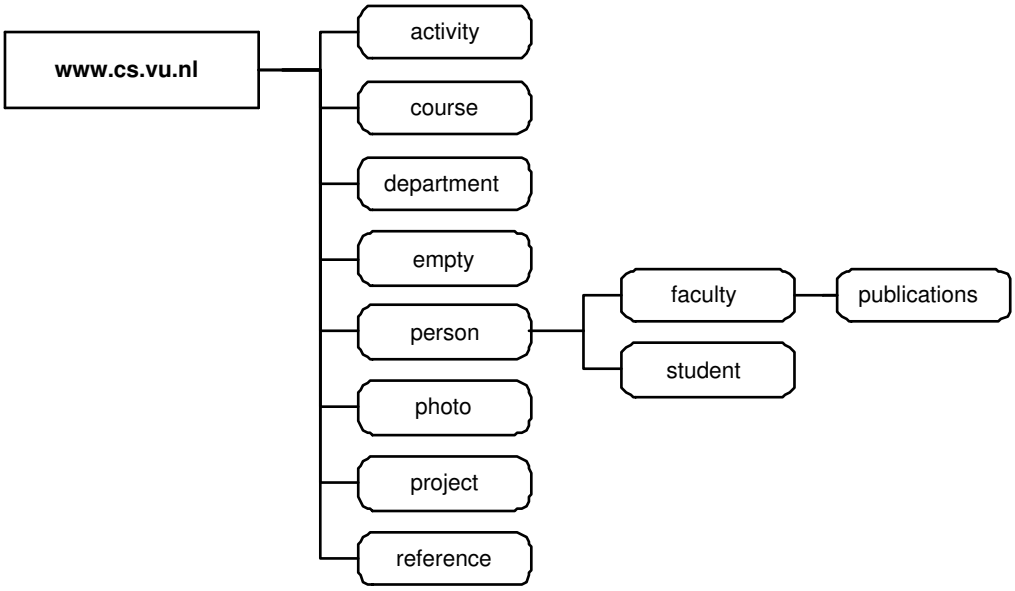

Figure 5.9: Target class structure used for classifying `www.cs.vu.nl`. Organization is based on the textual content of the page.

First, 100 documents were selected randomly for initial examples, then we used the lowest variance active learning method (described in Section 3.2.3) to select additional samples (50 more in each iteration). Aggregately 558 pages were analyzed and labeled by hand. Note that this means only the 4.2% of all documents. Finally 17 target categories contained examples.

5.4.4 Results

To obtain the accuracy of the classification we have picked out 328 pages randomly ($10 - 30$ from each category, depending on its size) and checked the target class values manually. According to these the accuracy was given to 74%. Table 5.2 contains the results of the verification.
The most difficult was to distinguish between department, faculty and project pages. We assume that the reason behind it is that these categories contain very similar words. The most unequivocally the student, reference, photo and empty pages were classified. The explanation is that these categories are more disjunct. More detailed results of the verification can be found in the Appendix, Table 7.10.
An other view-point is to consider just the language of the pages. In this interpretation there are three classes: English, Dutch and other language. The language was misclassified only 7 times from the 288 instances. It means 97.5% accuracy. In case of this three, almost[1] disjunct classes, the classifier can distinguish between the pages with really high accuracy.
It is informative to analyze the classification model. Words with highest probabilities effectively illustrate on which words the classification is based. We consider the

[1] English and Dutch are not fully distinct classes since these languages have some common words.

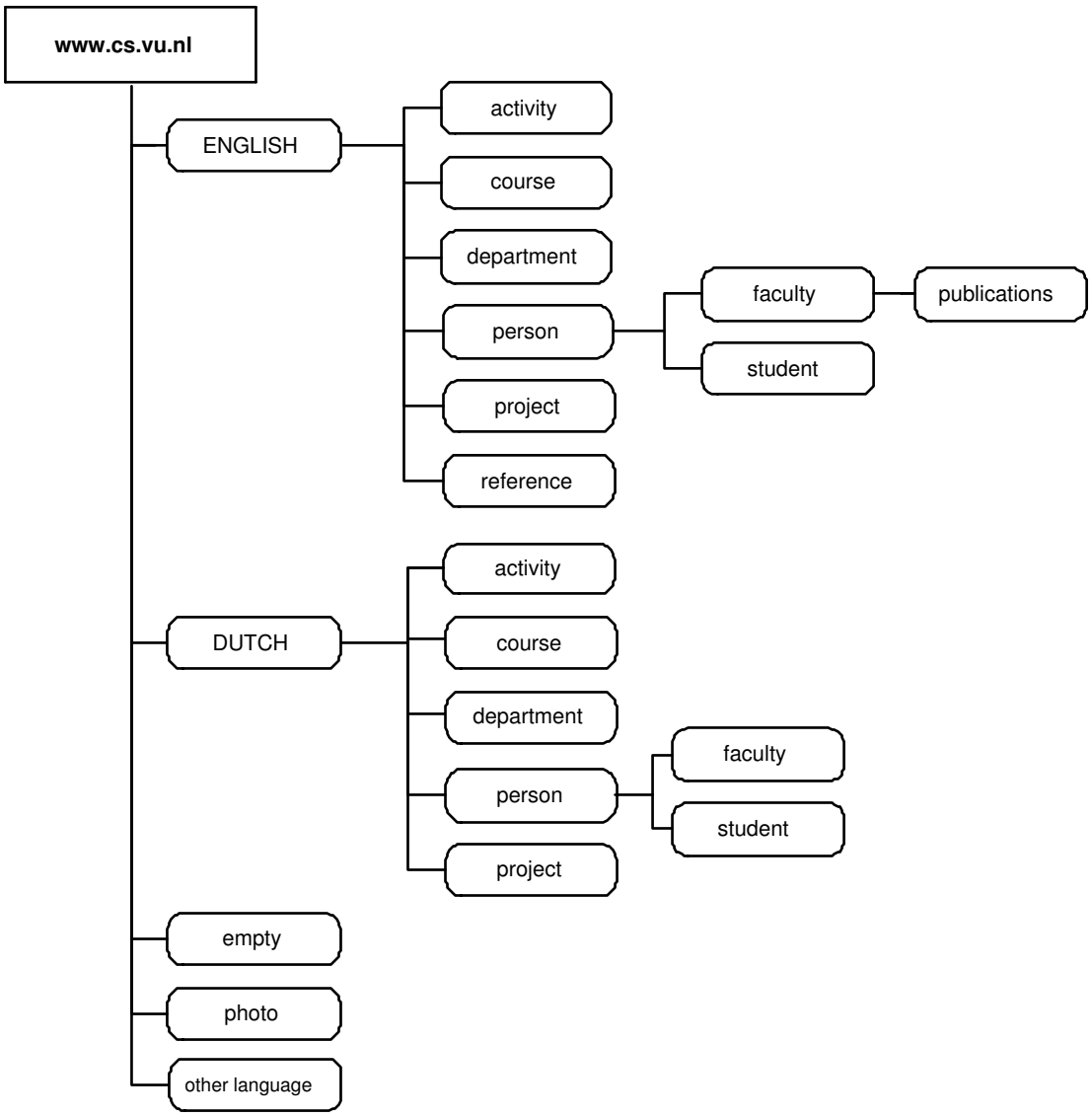

Figure 5.10: Target class structure used for classifying `www.cs.vu.nl`. Organization is based on the content and the language of the page.

category	tested	correct	percentage	wrong language
english/activity	20	12	60%	1
english/course	20	14	70%	0
english/department	20	11	55%	1
english/person/faculty	30	14	47%	3
english/person/faculty/publications	30	19	63%	0
english/person/student	20	18	90%	0
english/project	20	13	65%	0
english/reference	20	17	85%	0
dutch/activity	15	15	100%	0
dutch/course	20	15	75%	0
dutch/department	3	2	67%	1
dutch/person/faculty	10	6	60%	0
dutch/person/student	30	25	83%	0
dutch/project	10	5	50%	1
empty	20	17	85%	-
other language	20	20	100%	0
photo	20	19	95%	-
sum	328	242	74%	7

Table 5.2: Accuracy of the classification on randomly picked out pages. Wrong language column contains the number of documents for each target where the language was misclassified (English instead of Dutch and vice versa).

top 25 words with highest probabilities for each target. Photo category for example contains 'img', 'thumbnails', 'picture', 'foto' while English student pages contains 'me', 'my', 'about', 'music', 'birth', 'welcome' words. See Appendix 7.3 for more details.

Chapter 6
Conclusions

This paper has presented a family of algorithms that address the question of reducing the effort on example selection. This is an important question in text learning, because of the high cost of hand-labeling data. We have presented algorithms that improves classification accuracy and experimental results that show significant improvements in two real-world classification tasks.

In this paper we have shown that (1) reducing the size of the vocabulary can significantly increase performance, (2) using the richer structure of e-mail messages/html pages improves classification accuracy, (3) k-means clustering is a right alternative for selecting initial samples, (4) use of unlabeled data improves classification accuracy, (5) intelligent sample selection achieves reduces classification error, and (6) using prior user knowledge can improve classification accuracy. We have also described a method for producing rich feedback to users.

We see several other interesting directions for future work with these methods. K-means clustering performed well in initial sample selection, therefore it could be expanded to be used for sample selection. It pretends a faster implementation or the reduce of the document feature vector's size. In future work we intend to expand `iTM` with other active learning methods. User feedback also could be improved by visualizing more features of the classification model. We propose to investigate the different use of prior user knowledge.

Furthermore, other problem domains have similarities with text domains. Robotics, vision and information extraction are three such domains. Applying the techniques in this paper could improve performance in these areas as well.

Chapter 7
Appendix

7.1 Sample batch configuration file

We present a sample batch configuration file, used for classifying the 20-newsgroups data set, in the following. We neglect words below 100 and above 8000 frequency, and use the Naive Bayes classifier with the GMF active learning method. The first 100 samples are selected randomly then 16 times 175 by the active learner. The detailed and the average results of 10 runs are written into the `output.m` file. No unlabeled data and prior user knowledge are used.

```
runs = 10
sources = 20

%source type [TEXT/HTML/NNTP/EMAIL]
source[1][path] = data/20news-18828/alt.atheism
source[2][path] = data/20news-18828/comp.graphics
source[3][path] = data/20news-18828/comp.os.ms-windows.misc
source[4][path] = data/20news-18828/comp.sys.ibm.pc.hardware
source[5][path] = data/20news-18828/comp.sys.mac.hardware
source[6][path] = data/20news-18828/comp.windows.x
source[7][path] = data/20news-18828/misc.forsale
source[8][path] = data/20news-18828/rec.autos
source[9][path] = data/20news-18828/rec.motorcycles
source[10][path] = data/20news-18828/rec.sport.baseball
source[11][path] = data/20news-18828/rec.sport.hockey
source[12][path] = data/20news-18828/sci.crypt
source[13][path] = data/20news-18828/sci.electronics
source[14][path] = data/20news-18828/sci.med
source[15][path] = data/20news-18828/sci.space
source[16][path] = data/20news-18828/soc.religion.christian
source[17][path] = data/20news-18828/talk.politics.guns
source[18][path] = data/20news-18828/talk.politics.mideast
source[19][path] = data/20news-18828/talk.politics.misc
source[20][path] = data/20news-18828/talk.religion.misc
source[1][type] = NNTP
source[2][type] = NNTP
```

```
source[3][type] = NNTP
source[4][type] = NNTP
source[5][type] = NNTP
source[6][type] = NNTP
source[7][type] = NNTP
source[8][type] = NNTP
source[9][type] = NNTP
source[10][type] = NNTP
source[11][type] = NNTP
source[12][type] = NNTP
source[13][type] = NNTP
source[14][type] = NNTP
source[15][type] = NNTP
source[16][type] = NNTP
source[17][type] = NNTP
source[18][type] = NNTP
source[19][type] = NNTP
source[20][type] = NNTP
source[1][label] = alt.atheism
source[2][label] = comp.graphics
source[3][label] = comp.os.ms-windows.misc
source[4][label] = comp.sys.ibm.pc.hardware
source[5][label] = comp.sys.mac.hardware
source[6][label] = comp.windows.x
source[7][label] = misc.forsale
source[8][label] = rec.autos
source[9][label] = rec.motorcycles
source[10][label] = rec.sport.baseball
source[11][label] = rec.sport.hockey
source[12][label] = sci.crypt
source[13][label] = sci.electronics
source[14][label] = sci.med
source[15][label] = sci.space
source[16][label] = soc.religion.christian
source[17][label] = talk.politics.guns
source[18][label] = talk.politics.mideast
source[19][label] = talk.politics.misc
source[20][label] = talk.religion.misc

% vocabulary (0 to ignore an option)
voc_del_less = 100
voc_del_more = 8000
voc_max_size = 20000

%classifier [NaiveBayes/Boostexter]
```

```
classifier = NaiveBayes

example_iterations = 17
example_start_num = 100
example_increment = 175

%example selection [random/k-means]
example_selection = random

%use unlabeled data [0/1/2]
unlabeled = 0
unlabeled_size = 0

%intelligent sample selection (active learning) [0:no/1:CM/2:LW/3:GFM]
active = 3

%prior user knowledge
user = 0

output_file = output.m
```

7.2 20 newsgroups

7.2.1 Prior user knowledge

This enumeration contains a list of words, used as prior user knowledge in the 20-newsgroups data set. The list is created with the 'Representative words tool' of the iTM system.

alt.atheism:
agency, amusing, atheism, atheist, atheists, caused, gulf, islam, islamic, keith, moral, morality, penalty, political, rushdie, thoughts
comp.graphics:
format, gif, graphics, image, images, jpeg, library, polygon, pov, rumours, significance, tiff, viewer
comp.os.ms-windows.misc:
a86, b8f, bhj, bmp, challenge, cx, d9, deskjet, di, ei, g9v, giz, ini, max, microsoft, mouse, ms-windows, nt, os, pl, supporters, tm, um, windows
comp.sys.ibm.pc.hardware:
bios, bus, controller, cpu, eisa, gateway, ide, isa, motherboard, port, scsi, settings, vlb
comp.sys.mac.hardware:
apple, centris, duo, hours, iisi, kept, lc, lciii, mac, monitors, nubus, powerbook, quadra, se
comp.windows.x:

application, cursor, entry, escaped, motif, openwindows, r5, server, widget, widgets, window, x11, x11r5, xdm, xlib, xt, xterm, xv, xview

misc.forsale:

forsale, genesis, shipping

rec.autos:

auto, automotive, car, cars, concepts, dealer, dumbest, engine, ford, nissan, oil, saturn, self, shift, toyota, v12, v4, v6, v8, vx, warning

rec.motorcycles:

bike, bikes, bmw, chain, countersteering, dod, dog, dogs, motorcycle, ride, riding, shaft-drives, wave, wheelies

rec.sport.baseball:

baseball, braves, indians, jack, jays, morris, phillies, pitching, players

rec.sport.hockey:

abc, blues, bruins, buffalo, coverage, cup, detroit, devils, don, draft, espn, fans, game, goal, goalie, hockey, leafs, minus, montreal, nhl, pens, pittsburgh, played, playoff, pool, rangers, results, season, sharks, stat, team, teams, wc, wings

sci.crypt:

acceptance, agencies, algorithm, announcement, chip, clipper, code, considered, corporate, crypto, cryptography, db, des, encryption, escrow, hackers, hard-core, harmful, house, key, key-escrow, keys, keyseach, once, organized, pgp, privacy, screw, secret, secure, security, shelf, spooks, tapped, technical, tempest, white, wiretap

sci.electronics:

acid, batteries, circuit, concrete, copy, detector, detectors, disks, distance, electronics, ir, lead, phones, protected, signal, wiring

sci.med:

aids, bloom, cancer, candida, diet, disease, doctor, effects, fiction, homeopathy, krillean, medical, medicine, methodology, msg, photography, risk, sensitivity, superstition, tradition, treatment, yeast

sci.space:

billion, bursters, dc-x, gamma, henry, hst, launch, lunar, marketing, mars, mission, moon, nasa, orbit, planet, planets, ray, satellite, scheduled, servicing, shuttle, sky, solar, space, spacecraft, station, vandalizing

soc.religion.christian:

accepting, arrogance, bible, catholic, christ, christianity, christians, church, daily, doctrine, faith, god, god's, heart, heaven, hell, homosexuality, issues, jeesus, married, mary, mormon, rise, rutgers, sabbath, sin, verse

talk.politics.guns:

amendment, atf, batf, burns, control, dividian, express, fbi, fire, firearms, gun, guns, murders, ranch, survivors, waco, weapons

talk.politics.mideast:

arab, armenia, armenian, armenians, azerbaijan, bosnia, bosnians, europe, expansion, gaza, genocide, hezbollah, holocaust, israel, israeli, israelis, israel's, jews, muslim, muslims, nazi, peace, planes, shoot, soldiers,

solution, terrorism, turkey, turkish, turks

talk.politics.misc:

concentrate, cramer, employment, gay, govt, homosexuals, janet, limiting, molesters, nc, neighbor, percentage, stephanopoulos, study, tax, welcome

talk.religion.misc:

biblical, died, jesus, koresh's, loving, mormons, promise, who's

configuration	500	1500	2500	3500	4500	5500	6500
vocabulary 1	0.1779	0.4999	0.6841	0.7556	0.7859	0.8046	0.8191
vocabulary 2	0.2562	0.5797	0.7143	0.7619	0.7875	0.8034	0.8136
vocabulary 3	0.3539	0.6528	0.7293	0.7578	0.7758	0.7923	0.8008
vocabulary 4	0.4764	0.6564	0.7323	0.7514	0.7622	0.7672	0.7764

configuration	7500	8500	9500	10500	11500	12500
vocabulary 1	0.8226	0.8265	0.8338	0.8345	0.8375	0.8395
vocabulary 2	0.8239	0.8259	0.8321	0.8321	0.8324	0.8381
vocabulary 3	0.8071	0.8087	0.8079	0.8130	0.8137	0.8129
vocabulary 4	0.7816	0.7853	0.7883	0.7884	0.7858	0.7977

Table 7.1: Accuracy achieved by Naive Bayes classifier on the 20-newsgroups data set using different vocabulary configurations and random example selection

weight	100	450	800	1150	1500	1850	2200	2550	2900
0	0.1815	0.4112	0.5354	0.5943	0.6292	0.6551	0.6825	0.6899	0.7015
1	0.1829	0.4279	0.5661	0.6188	0.6637	0.6912	0.7109	0.7346	0.7371
10	0.1881	0.4951	0.6042	0.6676	0.7018	0.7369	0.7502	0.7669	0.7663
15	0.1954	0.4776	0.6046	0.6813	0.7091	0.7333	0.7489	0.7721	0.7710
20	0.1902	0.5132	0.5982	0.6588	0.7011	0.7268	0.7497	0.7581	0.7731

Table 7.2: Accuracy of the Naive Bayes classifier on the 20-newsgroups data set using different weights on the subject field of the news messages. Example documents are selected randomly.

7.3 Classifying www.cs.vu.nl

This section contains detailed results about the classification of the `www.cs.vu.nl` domain.

The following enumeration contains the top 25 relevant words found by the classifier for each target. Relevant in this context means that these words got the highest probabilities for the given target.

photo:
100, img, up, r1, thumbnails, 5, prev, januari, tim, afstudeerfeest, ilse, rolph, hush, picture, e-mail, phonebook, dejavu, foto, kerst, das, home, 4, arno, verdana, 13

empty:
wrl, dejavu, nosuch, dlp, www, rmi, server, online, untitled, w, last, port, apache/1, parent, slide, h, mar-2004, renambot, mijn, adriaan, jun-2004, note, slides, website, frames

dutch/department:
universiteit, vrije, sciences, faculty, few, studenten, ftp, theoretische,

method	100	450	800	1150	1500	1850	2200	2550	2900
random	0.1881	0.4951	0.6042	0.6676	0.7018	0.7369	0.7502	0.7669	0.7663
CM 8	0.1967	0.4138	0.5085	0.6080	0.6519	0.6821	0.7021	0.7148	0.7256
LW 8	0.2200	0.4744	0.5577	0.6168	0.6645	0.6993	0.7308	0.7414	0.7572
LW 16	0.2125	0.4449	0.5443	0.6113	0.6602	0.6991	0.7268	0.7516	0.7706
GMF 8	0.1774	0.3299	0.4637	0.5647	0.6303	0.6826	0.7227	0.7544	0.7730
GMF 16	0.2457	0.3959	0.5267	0.6155	0.6775	0.7116	0.7439	0.7702	0.8010

Table 7.3: Accuracy of the Naive Bayes classifier on the 20-newsgroups data set using different active learning methods. 'Random' denotes random example selection while 'CM', 'LW' and 'GMF' denoting the corresponding active learning method. The number behind the method's name denotes the number of iterations.

method	100	450	800	1150	1500	1850	2200	2550	2900
random	0.1881	0.4951	0.6042	0.6676	0.7018	0.7369	0.7502	0.7669	0.7663
LW	0.3147	0.4740	0.5516	0.6144	0.6616	0.7030	0.7360	0.7578	0.7778
GMF	0.2887	0.4030	0.5182	0.5981	0.6606	0.7085	0.7400	0.7686	0.7913

Table 7.4: Accuracy of the Naive Bayes classifier on the 20-newsgroups data set using different initial sample selection methods. 'Random' denotes random example selection while 'LW' and 'GMF' denoting the corresponding active learning method with k-means clustering for initialization.

labeled	0	500	1000	2000	5000	10000	12000
100	0.1911	0.1881	0.1702	0.2089	0.2130	0.3358	0.2521
1000	0.6376	0.6273	0.6173	0.6491	0.6873	0.6934	0.7038
1900	0.7369	0.6947	0.7028	0.7102	0.7325	0.7478	0.7507
2800	0.7663	0.7375	0.7349	0.7424	0.7574	0.7788	0.7733

Table 7.5: Accuracy of the Naive Bayes classifier on the 20-newsgroups data set while varying the amount of unlabeled data. Example documents are selected randomly.

selection	100	450	800	1150	1500	1850	2200	2550	2900
random	0.1881	0.4951	0.6042	0.6676	0.7018	0.7369	0.7502	0.7669	0.7663
HS	0.2065	0.5636	0.6674	0.7056	0.7411	0.7492	0.7529	0.7713	0.7725
HV	0.1780	0.5341	0.6762	0.7098	0.7440	0.7482	0.7675	0.7767	0.7758

Table 7.6: Accuracy of the Naive Bayes classifier on the 20-newsgroups data set using different unlabeled data selection methods. Example documents are selected randomly.

weight	100	450	800	1150	1500	1850	2200	2550	2900
0	0.1881	0.4951	0.6042	0.6676	0.7018	0.7369	0.7502	0.7669	0.7663
3	0.2350	0.5335	0.6461	0.6848	0.7065	0.7257	0.7514	0.7543	0.7622
5	0.2578	0.5714	0.6532	0.6790	0.7031	0.7243	0.7440	0.7490	0.7496
10	0.3124	0.5609	0.6138	0.6571	0.6735	0.6918	0.6998	0.7004	0.7088

Table 7.7: Accuracy of the Naive Bayes classifier on the 20-newsgroups data set using different weights on prior user knowledge. Example documents are selected randomly.

method	100	450	800	1150	1500	1850	2200	2550	2900
random	0.1881	0.4951	0.6042	0.6676	0.7018	0.7369	0.7502	0.7669	0.7663
all features	0.4207	0.5542	0.6136	0.6519	0.6782	0.7041	0.7174	0.7274	0.7344

Table 7.8: Comparison of random selection and the use of all `iTM` features. 'All features' includes k-means clustering for initialization, intelligent sample selection using GMF, use of unlabeled data (10000 documents), and prior user knowledge.

huidige, naar, inf, e-mail, contact, kunstmatige, foto's, amsterdam, english, adres, informatica, site, nederland, introductie, boelelaan, webmaster, browser

english/reference:

genericq, typeable, generic, haskell, type, ppt, forall, module, private, javax, swing, java, generics, html, types, online, content, version, package, evolution, key, functions, cell, function, introduction

english/activity:

tel, workshop, ontologies, amsterdam, vrije, conference, committee, home, university, universiteit, applications, international, science, csapc, link, ontology, methods, department, here, fi, theory, please, part, papers, faculty

english/department:

click, right, broadband, faculty, modem, vrije, universiteit, sciences, mb, download, ph, e-mail, scientific, phonebook, section, term, distributed, students, amsterdam, dr, department, staff, our, parallel, does

english/project:

parallel, project, computing, wide-area, vrije, albatross, applications, universiteit, communication, wide, kielmann, henri, bal, thilo, area, support, java, dog, workshop, amsterdam, distributed, cluster, rdf, das, agent

english/person/faculty:

home, science, vrije, universiteit, faculty, dr, amsterdam, com, homepage, books, search, languages, dick, international, online, internet, mathematics, university, personal, project, sciences, parsing, free, compiler,

group

english/person/student:

me, my, about, music, jul, jennifer, rome, baby, birth, apr, q, go, aug, oct, welcome, capri, linda, adoptee, does, richard, mary, take, how, don't, she

english/person/faculty/publications:

ps, gz, zip, h, steen, distributed, proc, approach, paper, g, based, f, maintenance, study, l, analysis, architecture, case, model, security, our, such, hofman, globe, technical

english/course:

do, ps, faculty, gz, define, msg, universiteit, vrije, latex, science, course, virtual, make, home, e-mail, malloc, learning, like, scientific, business, names, phonebook, machine, parallel, sciences

dutch/course:

5, 0, wi, 4000160, 24-mei-2004, 4, die, 6, dat, -, als, er, 7, adobe, college, deze, y, trs, niet, informatie, door, mensen, bij, museum, worden

dutch/person/student:

mijn, ik, homepage, website, je, naar, links, hier, site, math, terug, kijk, browser, ben, mail, deze, aan, dat, mij, welkom, vrije, everybody, untitled, universiteit, uitleg

dutch/person/faculty:

hoofdstuk, 0, wiskunde, poster, onderwijs, e-mail, km, koole, appendix, bwi, home, v, uva, ps, link, write, view, faculteit, mathematics, format, amsterdam, ger, universiteit, boelelaan, then

other language:

di, del, la, della, per, agrave, il, con, le, art, dei, da, cui, un, que, chiavi, che, delle, sono, se, al, decreto, si, ed, su

dutch/project:

few, requirements, ik, kan, ook, dan, dat, die, wil, engineering, zijn, hoe, support, er, maar, worden, dit, product, multi-agent, als, eacute, over, hij, je, gebruiker

dutch/activity:

storm, je, lustrum, symposium, nieuws, augustus, uuml, bij, over, deze, die, komt, hier, zijn, dag, door, maar, maken, tijdens, wie, kan, dit, poster, studenten, ding

category	examples		classified		all pages	
photo	98	18%	2334	19%	2432	19%
empty	209	37%	2518	20%	2727	21%
dutch/department	1	0%	2	0%	3	0%
english/reference	32	6%	934	8%	966	7%
english/activity	6	1%	26	0%	32	0%
english/department	15	3%	254	2%	269	2%
english/project	23	4%	418	3%	441	3%
english/person/faculty	35	6%	1049	8%	1084	8%
english/person/student	23	4%	526	4%	549	4%
english/person/faculty/publications	25	4%	1876	15%	1901	15%
english/course	10	2%	101	1%	111	1%
dutch/course	10	2%	796	6%	806	6%
dutch/person/student	47	8%	1163	9%	1210	9%
dutch/person/faculty	7	1%	3	0%	10	0%
other language	9	2%	408	3%	417	3%
dutch/project	4	1%	23	0%	27	0%
dutch/activity	4	1%	22	0%	26	0%
sum	558		12453		13011	

Table 7.9: Document frequencies for each target class of `www.cs.vu.nl`.

category	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
0	19	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
1	-	17	-	-	1	-	-	-	-	-	-	-	1	1	-	-	-
2	-	-	2	-	-	-	1	-	-	-	-	-	-	-	-	-	-
3	-	1	-	17	1	-	-	-	-	-	1	-	-	-	-	-	-
4	-	1	-	1	12	-	1	1	1	2	-	-	-	1	-	-	-
5	1		1	-	-	11	3	3	-	-	1	-	-	-	-	-	-
6	-	1	-	1	-	-	13	3	-	2	-	-	-	-	-	-	-
7	1	2	2	-	2	2	-	14	1	4	1	-	-	1	-	-	-
8	1	-	-	-	-	-	-	-	18	1	-	-	-	-	-	-	-
9	-	-	-	4	1	1	2	-	-	19	3	-	-	-	-	-	-
10	-	1	-	4	-	-	-	1	-	-	14	-	-	-	-	-	-
11	1	-	-	-	-	-	-	-	-	-	-	15	2	-	-	-	2
12	1	1	2	-	-	-	-	-	-	-	-	-	25	-	-	-	1
13	1	-	-	-	-	-	-	-	-	-	-	3	-	6	-	-	-
14	-	-	1	1	-	-	-	-	-	-	-	3	-	-	20	-	-
15	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	5	-
16	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	15

Table 7.10: Detailed result on validating the classification of `www.cs.vu.nl`. The corresponding category labels are the following: 0: photo, 1: empty, 2: dutch/department, 3: english/reference, 4: english/activity, 5: english/department, 6: english/project, 7: english/person/faculty, 8: english/person/student, 9: english/person/faculty/publications, 10: english/course, 11: dutch/course, 12: dutch/person/student, 13: dutch/person/faculty, 14: other language, 15: dutch/project, 16: dutch/activity.

Bibliography

[1] Shlomo Argamon-Engelson and Ido Dagan. Commitee-based sample selection for probabilistic classifiers. *Journal of Artificial Intelligence Research*, (11):335–360, 1999.

[2] Ron Bekkerman, Ran El-Yaniv, Naftali Tishby, and Yoad Winter. Distributional word clusters vs. words for text categorization. *Journal of Machine Learning Research*, (3):1183–1208, 2003.

[3] Lewis D.D and Knowles K.A. Threading electronic mail: A preliminary study. *Information Processing and Management*, 33(2), 209-217, 1997.

[4] Lewis D.D. and Gale W.A. A sequential algorithm for training text classifiers. *SIGIR '94: Proceedings of the Seventeenth Annual International ACM SIGIR Conference on Research and Developement in Information Retrieval*, pp. 3-12, 1994.

[5] Inderjit S. Dhillon, Subramanyam Mallela, and Rahul Kumar. A divisive information-theoretic feature clustering algorithm for text classification. *Journal of Machine Learning Research*, (3):1265–1287, 2003.

[6] George Forman. An extensive empirical study of feature selection metrics for text classification. *Journal of Machine Learning Research*, (3):1289–1305, 2003.

[7] Johannes Frnkranz, Tom Mitchell, and Ellen Riloff. A case study in using linguistic phrases for text categorization on the WWW.

[8] Rayid Ghani, Rosie Jones, Dunja Mladenic, Kamal Nigam, and Sean Slattery. Data mining on symbolic knowledge extracted from the Web.

[9] Peter I. Hofgesang. Web usage mining (structuring semantically enriched clickstream data).

[10] Lang K. Newsweeder: Learning to filter netnews. *Machine Learning: Proceedings of the Twelfth International Conference (ICML '95)*, pp. 331-339, 1995.

[11] David Landau, Ronen Feldman, Yonatan Aumann, Moshe Fresko, Yehuda Lindell, Orly Lipshtat, and Oren Zamir. TextVis: an integrated visual environment for text mining.

[12] Craven M., DiPasquo D., Freitag D, McCallum A., Mitchell T., Nigam K., and Slattery S. Learning to extract symbolic knowledge from the world wide

web. *Proceedings of the Fifteenth National Conference on Artificial Intelligence (AAAI-98)*, pp. 509-516.

[13] Tom M. Mitchell. *Machine Learning.* The McGraw-Hill Companies, Inc., 1997.

[14] Tom M. Mitchell. The role of unlabeled data in supervised learning. *Proceedings of the Sixth International Colloquium on Cognitive Science*, 1999.

[15] Pazzani M.J., Muramatsu J., and Billsus D. Indentitying interesting Web sites. *Proceedings of the Thirteenth National Conference on Artificial Intelligence (AAAI-96)*, pp. 54-59, 1996.

[16] Dunja Mladenic. *Machine Learning on non-homogeneous, distributed text data.* PhD thesis, University of Ljubljana, 1998.

[17] Kamal Nigam, Andrew K. McCallum, Sebastian Thrun, and Tom Mitchell. Text classification from labeled and unlabeled documents using EM. *Machine Learning*, Kluwer Acedemic Press, 1999.

[18] Jorg Ontrup, Tim W. Nattkemper, Olaf Gerstung, and Helge Ritter. A MeSH term based distance measure for document retrieval and labeling assistance.

[19] Choon Yang Quek. Classification of World Wide Web documents. Master's thesis, School of Computer Science, CMU.

[20] Gerard Salton and Christopher Buckley. Term-weighting approaches in automatic text retrieval. *Inf. Process. Manage.*, 24(5):513–523, 1988.

[21] Robert E. Schapire and Yoram Singer. Boostexter: A boosting-based system for text categorization. *Machine Learning*, 39(2/3):135–168, 2000.

[22] Joachims T. Text Categorization With Support Vector Machines: Learning with many relevant features. *Machine Learning: ECML-98, Tenth European Conference on Machine Learning*, pp. 137-142, 1998.

[23] Yiming Yang and Jan O. Pedersen. A comparative study on feature selection in text categorization. In Douglas H. Fisher, editor, *Proceedings of ICML-97, 14th International Conference on Machine Learning*, pages 412–420, Nashville, US, 1997. Morgan Kaufmann Publishers, San Francisco, US.