Relevance Feedback using Support Vector Machines

Harris Drucker HD@RESEARCH.ATT.COM
AT&T Research and Monmouth University, West Long Branch, NJ 07764, USA
Behzad Shahrary BEHZAD@RESEARCH.ATT.COM
David C. Gibbon DCG@RESEARCH.ATT.COM
AT&T Research, 200 Laurel Ave. Middletown, NJ, 07748, USA.

Abstract

We show that support vectors machines (SVM's) are much better than conventional algorithms in a relevancy feedback (RF) environment in information retrieval (IR) of text documents. We track performance as a function of feedback iteration and show that while the conventional algorithms do very well in the initial feedback iteration if the topic searched for has high visibility in the data base, they do very poorly if the relevant documents are a small percentage of the data base. SVM's however do very well when the number of documents returned in the preliminary search is low and the number of relevant documents is small. The competitive algorithms examined are Rocchio, Ide regular, and Ide dec-hi.

1. Introduction

1.1 Statement of the problem

Our problem is that of relevancy feedback (RF) within the context of information retrieval (IR). There is a set of documents that a user wants to retrieve within a database. Some of the articles are relevant, some not. It is important to understand that relevancy is relative to the perception of the user, that is document D_j may be relevant to user U_k but not user U_p.

The user is assumed to present a preliminary (or initial) query to the system, in which case the system returns a set of ranked documents which the user examines. Although many documents may be retrieved by the system, the system only presents one screen of documents at a time. In our case, we assume that ten documents are returned on the initial screen with enough information for the user to gauge whether a document is relevant or not. An optimal preliminary query would only return relevant documents and all the user has to do is scroll through all the screens to find all the relevant documents. In actuality, depending on the quality of the initial query, many documents may be returned but few may be relevant. The initial query may be a Boolean query such as conjunction or disjunctions of key words or the inquiry could be a sophisticated inquiry in the form of a question.

In our case, we ignore the exact nature of the preliminary query and assume that the return of the documents from this initial query is poor (three or less relevant documents from the full screen of ten documents). However, our technique will work no matter how many documents are returned from the initial query. We believe that a hard test of an IR system with RF occurs when the number of relevant documents returned in the initial query is low. We assume that if the documents returned in the initial screen are all relevant, then the user will just scroll to the next screen while if no relevant documents are returned, the user continues to scroll through the screens until at least one relevant document is returned on a screen and then the first feedback iteration begins. Thus if there are between one and nine relevant documents returned on the initial screen, the user marks the relevant documents (unmarked documents are taken as non-relevant) and the system goes though a first feedback iteration and another set of the top ten ranked documents are returned. These feedback iterations continue until the user terminates the procedure.

We first concentrate on the state when between one and nine (inclusive) relevant documents are returned in the initial screen. Our method will be based on the use of support vector machines (Vapnik, 1998; Drucker, 1999; Joachims, 1998) with comparisons to other RF techniques: Rocchio (1971), Ide regular and Ide dec-hi (Salton and Buckley, 1990; Harman, 1992). These algorithms will be examined in detail later but suffice to say now that all except Ide dec-hi use all the relevant and non-relevant documents on the first screen while Ide dec-hi uses all the relevant documents and the top ranked non-relevant document.

Recall that we are paying particular attention to the case where the initial retrieval is poor. As anyone who has done IR or web searches will attest, it is rather discouraging to get a return of a search stating that the search has found thousands of documents when in fact most of the documents on the first screen (the highest ranked documents) are not relevant to the user. Our typical user is hypothesized as preferring to mark the top ten returned documents as relevant or not and going through a series of feedback iterations rather that examining many screens to mark all the relevant documents.

Summarizing: in the initial preliminary search we obtain either (1) no relevant documents, (2) one to nine relevant documents or (3) all relevant documents. In case (1), we will be forced to go to succeeding screens until we get one screen with at least one relevant document. All the documents on that last screen and previous screens will be used in the first feedback iteration. In case (2) we mark the relevant documents on the first screen (unmarked on the first screen are non-relevant) and go through feedback iterations. In case (3) we go to the next screen. We will not investigate case (3). In our case we will concentrate on the situation when the number of relevant documents returned on the first screen is low (three or less) and could be zero. Please distinguish between the preliminary query that returns the first set of documents and the first feedback iteration which starts with that initial set of documents marked by the user.

After the first feedback iteration and on all subsequent iterations we will examine only the first screen no matter how many of the returned documents are relevant (even if none). We then track performance as a function of feedback iteration.

1.2 Techniques Investigated.

One difference between our study and others is the simultaneous tracking of performance as a function of feedback iteration and the use of SVM's. Although there have been many studies of the use of SVM's in text retrieval (Joachims, 1998; Vapnik, 1998, 1992; Drucker, 1999), most studies emphasize finding the method that optimizes performance after one feedback iteration.

SVM's have been studied in the context of the IR filtering problem (Dumais, 1998; Joachims, 1998). It is understood that both RF and filtering problems are both classification problems in that documents (in our case) are assigned to one of two classes (relevant or not). However, in the filtering situation, we usually have a marked set of documents termed the training set and use that set to train a classifier. Performance is then judged on a separate test set. In a sense, filtering could be considered RF with only one feedback iteration. The problem with using filtering rather than many iterations of RF is that (1) one has to mark "many" documents in the training set to obtain reasonable classification rates on the test set (2) how many is "many" depends on the problem and is not known in advance (3) since what are to be considered relevant documents is user dependent, this would mean that every user must construct a different training set. Multiple iterations of feedback could be considered to be an attempt to maximize performance on a test set that includes all documents except the ones already marked. In that sense, RF is similar to what is termed active learning (Tong, 2000; Scholhn and Cohen, 2000) in that we try to maximize test performance using the smallest number of documents in the training set. However the important difference between RF and active learning is that active learning may force the user to mark many more non-relevant documents than in IR feedback and our supposition is that the user wants to see the maximum number of relevant documents at each feedback iteration. Additionally, in active learning we are interested in maximizing performance on the entire test set. In our case we are interested in maximizing performance on the next ten documents retrieved.

IR and RF have a long history. In the Rocchio (1971) algorithm formulation we have a set of documents, each document represented by a vector $\mathbf{D_i}$, the elements of which we will discuss later. The preliminary query (not necessary returned by a Rocchio feedback iteration) contains N total documents. If the preliminary search realizes between one and nine relevant documents (and the resultant number of non-relevant documents), then N is 10. If there are no relevant documents on the first screen than we search subsequent screens until there is at least one relevant document – in this case N is a multiple of ten. We will ignore the case of ten relevant documents returned on the preliminary search because then the preliminary query is very good and one just goes to subsequent screens to retrieve more relevant documents.

The feedback iteration using Rocchio after the initial (non-Rocchio) search forms the following query:

$$\mathbf{Q}_j = a\mathbf{Q}_{j-1} + \frac{b}{R}\sum_{\text{Relevant}} \mathbf{D}_i - \frac{g}{N-R}\sum_{\text{Non-Relevant}} \mathbf{D}_i$$

where $j=1$ for the initial iteration, $\mathbf{Q}_0 = 0$, R and N-R are the number of relevant and non-relevant documents, respectively, retrieved on the present iteration. We defer the description of the elements of **D** to later except to say here that **D** is normalized to unit length and negative elements of **Q** set to zero (rationale for this is in Rocchio (1971)). To implement the first iteration we form the dot product of this first query against all the documents $(\mathbf{Q_1} \bullet \mathbf{D_i})$ where the documents are those not yet marked as relevant and non-relevant and then we rank the dot products from high to low, present the ten largest dot products for the user to mark as relevant and non-relevant and then continue to the next iteration.

After the first relevancy iteration, we form subsequent iterations using the old value of **Q** and the relevant and non-relevant documents identified in the previous iteration.

We have a number of concerns with the Rocchio algorithm that we feel will make it problematic for use; mainly that it depends on three constants (a, b, g). Most studies on the Rocchio algorithm vary the three constants to determine the optimum choice of these constants. However, we feel that is unfair – a separate validation set should have been used. Furthermore, even if one has a validation set, one does not have time to search for that optimum set of constants. Thus, we will use the following choices of a, b, g as 8, 16, and 4 respectively, a choice that seemed to work well elsewhere (Buckley, Salton & Allan, 1994). Since all the negative elements of the resultant query are set to zero and since we are assuming that most of the documents returned in the next iteration will be non-relevant, there will be many elements of the query set to zero making for very poor performance on the next iteration.

The Ide regular algorithm (Salton & Buckley, 1990; Harman, 1992) is of the following format:

$$\mathbf{Q}_j = \mathbf{Q}_{j-1} + \sum\nolimits_{\text{Relevant}} \mathbf{D}_i - \sum\nolimits_{\text{Non-Relevant}} \mathbf{D}_i$$

where for the first feedback iteration, the **Q** on the right hand side is zero and as usual, all negative elements of the resultant query are set to zero.

The Ide dec-hi has basically the same form as Ide regular except the last summation has only one term, namely the highest ranked non-relevant document.

The fourth technique will be based on the use of support vector machines. SVM's can best be understood in the context of Figure 1 where the black diamonds represents the relevant vectors **D** in high dimensional space and the empty diamonds represent the non-relevant documents.

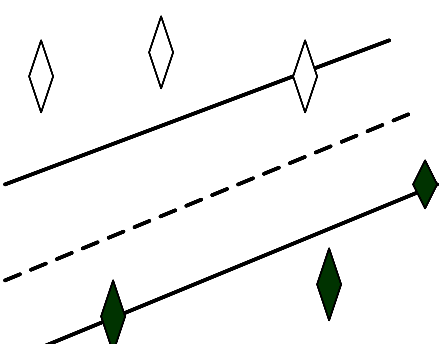

Figure 1. Support vectors and separating hyperplanes.

When SVM's are constructed, two sets of hyperplanes are formed (the solid lines), one hyperplane going through one or more examples of the non-relevant vectors and one hyperplane going through one or more examples of the relevant vectors. Vectors lying on the hyperplanes are termed support vectors and in fact define the two hyperplanes. If we define the margin as the orthogonal distance between the two hyperplanes, then a SVM maximizes this margin. Equivalently, the optimal hyperplane is such that the distance to the nearest vector is maximum. Details for solving this problem may be obtained elsewhere (Joachims, 1998; Vapnik, 1992, 1998). Here we use the vector **Q*** instead of the traditional **w** for the optimal hyperplane. **Q*** is such that the margin $2/|\mathbf{Q}^*|$ is maximized by finding the a's that maximizes:

$$W(a) = \sum_{i=1}^{N} a_i - .5 \sum_{i=1}^{N} \sum_{j=1}^{N} a_i a_j (\mathbf{D_i} \bullet \mathbf{D_j}) y_i y_j$$

where $0 \le a_i \le C, \quad y_i = \pm 1, \quad \sum y_i a_i = 0$. Then $\mathbf{Q}^* = \sum\nolimits_{i=1}^{r} a_i y_i \mathbf{D}_i$ and the sum is over the *r* support vectors $(a_i \neq 0)$.

The advantage of the linear representation is that **Q*** can be calculated after training and classification amounts to computing the dot product of **Q*** against every new document vector.

Linear SVM's execution speeds are very fast and there is only one parameter to tune (*C*), which is a restriction on the largest value of a. Another advantage of SVM's is that they are remarkably intolerant of the relative sizes of the number of training examples of the two classes. In most learning algorithms, if there are many more examples of one class than another, the algorithm will tend to correctly classify the class with the larger number of examples, thereby driving down the error rate. Since SVM's minimize the error rate by attempting to separate patterns in high dimensional space, the result is that SVM's are relatively insensitive to the number in each class.

In our model of RF, after construction of **Q*** we calculate $\mathbf{Q}^* \bullet \mathbf{D_i}$ for all documents **not** seen so far and rank them from high to low (assuming the relevant documents are of class +1) and return the top ten to the user for marking. These values represent the proportional distances from the optimal separating hyperplane to the vectors of the documents not used so far in constructing the optimal hyperplane. These documents may be inside or outside the margin (since they were not used to generate the present support vectors). Those documents on the class +1 side of the optimal hyperplane and furthest from the optimal hyperplane are the top ranked documents. Some of these top-ten ranked documents may in fact be non-relevant and in the next feedback iteration these newly marked vectors (in addition to those marked in previous feedback iterations) are used to construct a new set of support vectors. We contrast this with active learning (Schohn and Cohen, 1998; Tong and Keller, 2000) where the emphasis will be to take vectors in the next feedback iteration from within the margin. We don't want to do this in RF as many of the points within the margin will be non-relevant and not useful to the user. If the top-ten ranked documents are outside the margin and are all relevant, then in the next feedback iteration the support vectors will not change. If any of the top ten documents are within the margin, the next set of support vectors will be different from the present set.

Solving the previous set of equations is done using SVM^{light} (see Acknowledgement section). There are not many candidate vectors to consider as support points. In general the number of potential support points is ten times the iteration number.

1.3 Related Work on SVM's, IR, and RF

Harman (1992) compares many models including probabilistic models (as opposed to vector space models) and is the only paper we could find that tracks performance as a function of iteration. Schapire, Singer, and Singhal (1998) investigate a modified Rocchio and boosting as applied to text filtering. Although their investigation was not in the RF domain, they did show that a modified Rocchio algorithm could do much better than the original Rocchio algorithm. However, their algorithm requires multiple passes over documents and is problematic for large text collections.

The article by Lewis, Schapire, Callan, & Papka (1996) is in the area of text classification and compares Rocchio with Woodrow-Huff and exponentiated gradient. Joachims (1997) compared Rocchio and naïve Bayes in text categorization while Salton and Buckley (1990) examine Rocchio and probabilistic feedback but only for one feedback iteration. They use the residual collection method (as do we) in that performance is measured on all those documents in the collection not yet used in a feedback iteration.

Joachims (1998) looked at SVM's in text categorization and compared this to naïve Bayes, C4.5, k-nearest neighbor, and Rocchio. Although not a RF study, it discusses the issue of whether all or just some of the features should be used (features are the elements of the vectors). Although reducing the number of features does improve performance on some algorithms (k-nearest neighbor, C4.5 and Rocchio), it does not for naïve Bayes and SVM. It is our contention that we cannot waste time searching for the best set of features and so we use the full set of features in our study.

Buckley, Salton, and Allen (1993) examined IR within the context or a routing problem. They use the Rocchio algorithm modified so that the last term in the equation defining the new query includes not only the non-relevant documents marked on the present screen but assumes that all unseen documents are non-relevant and included in the last term. The same three authors (1994) also examined the use of locality information to improve performance.

In all these discussions of relevance feedback, it is often assumed that the preliminary query is of high quality. We make no such assumptions here.

2. Term Weighting Issues

We discuss the issue of the term t_i in the document vector $\mathbf{D}_j$. In the IR field, this term is called the term weighting while in the machine learning field, this term is called the feature and in linear algebra it is the i'th element of the vector $\mathbf{D}_j$. t_i states something about word i in document $\mathbf{D}_j$. If this word is absent, t_i is zero. If the word is present, then there are several options. One option is that the term weight is a count of the number of times this word occurs in this document (called the term frequency (TF)). The next option is that this term just indicates whether this word is present or not (binary term weighting). In the original Rocchio algorithm, each term TF is multiplied by a term $\log(N/n_i)$ where N is the total number of documents in the collection and n_i is the number of documents in which this term occurs. This last term is called the inverse document frequency (IDF). Usually $\mathbf{D}_j$ is normalized to unit length.

A popular combination for feature i is the multiplication of the TF by the IDF (TF-IDF). Salton and Buckley (1988) discuss in detail various term weighting options. Schapire, Singal, and Singer (1998) also look at different term weighting options. However, using TF-IDF requires two passes over the data because IDF cannot be determined until all the words and the number of articles in which that word is present is calculated. A compensating fact is that these two passes only have to be done once for any static collection of documents. Although we will evaluate algorithms using TF-IDF we would prefer not to use it in practice. The question will be whether using TF-IDF is much better than using just TF. Buckley, Salton, and Allen (1994) use TF-IDF for the preliminary query but not for the documents themselves since the determination of IDF is quickly done for the small numbers of queries as opposed to the large number of documents.

3. Stemming and Stop Lists.

Full stemming is the removal of suffixes of a word. For example, "builder", "building", and "builds" will all be reduced to their common root "build". There could also be partial stemming such as changing plural forms to their singular. Stemming reduces the size of the document vectors. One performance issue will be the effect of stemming on retrieval accuracy. But there are other performance issues such as retrieval speed and size of the inverted index. Buckley, Salton and Allen (1993, page 68), discuss these options in detail.

Another technique to reduce the dimensionality is the removal of "stop" words, that is, a list of words that will not be used in constructing the document vector. The use of a stop list may or may not improve performance. Words like "a", "an", and "the" probably do not help in determining relevancy. If the stop list consists of an a priori list of words, then only one pass over the documents is needed. However, if the stop list is based on the proportion of documents that has certain words, then two passes are required over the documents – once to count the number of all words and the second pass to eliminate either very common words or very rare words. One version of this is to remove words that do not occur in at least three documents (Drucker, 1999). This eliminates rare or misspelled words.

We tried the following combinations of pre-processing techniques and algorithms (1) Stemming using the Porter (1980) stemmer or not. (2) TF, TF-IDF, or binary features (3) Four algorithms: Rocchio, Ide regular, Ide deci-hi and SVM's. Number 1 can be done "on the fly" but TF-IDF of number 2 requires two passes over the document. In all cases, all one or two letter words were eliminated, words reduced to lower case, and the words "and" and "the" were eliminated.

We did not try all combinations of features because previous studies (Drucker, 1999) had shown that TF-IDF is not necessary for SVM's Salton and Buckley (1988) showed that using binary features gives the worst performance in comparison to TF or TF-IDF. Recall that we would prefer not to use TF-IDF because it requires two passes to calculate IDF.

4. Performance Metrics.

There are too many ways to assess the effectiveness of the feedback process to discuss here in detail. References are Lewis (1995), Tauge-Sutcliffe (1992), Saracevic (1975), Mizzaro (1997), and Korfhage (1997). However, traditionally recall and precision are used. Let R be the number of relevant documents in the collection, n_{Rel} be the number of relevant documents actually retrieved in a feedback iteration and N be the total number of documents returned in the feedback iteration (typically, N is 10). Precision (p) and recall(r) are then defined as:

$$p = n_{\text{Rel}} / N \qquad r = n_{\text{Rel}} / R$$

Although we assume N is ten if one through nine relevant documents are returned on the initial preliminary search, if we did change N, both the recall and precision will change (because n_{Rel} does) and at some point both the recall and precision will be approximately equal and this is termed the recall-precision break-even point and is a popular measure of performance. Schapire, Singer, and Singhal (1998) give very reasonable arguments why conventional metrics such as the precision-recall break-even point are not very informative to the user. In particular, we concur with their contention that since recall cannot be calculated by the user until all relevant documents are seen by the user (if this is even possible except in an exhaustive search), the user cannot know (in terms of recall) how well the IR search is going.

For these reasons, we will emphasize the coverage ratio as the performance metric. Coverage ratio is a cumulative metric and is the ratio of the cumulative total number of relevant documents retrieved so far to the cumulative number of documents that would have been retrieved in an ideal search. The coverage ratio is ideally unity at each iteration.

To take account of the fact that at some point an ideal RF system will run out of relevant documents to retrieve, we define the coverage ratio as:

$$\text{coverage ratio} = \begin{cases} \dfrac{\sum n_{R_i}}{10i} & \text{when } 10i \le R \\ \dfrac{\sum n_{R_i}}{R} & \text{otherwise} \end{cases}$$

where R is the total number of relevant documents in the entire collection and n_{R_i} is the number of relevant documents returned at iteration i. The user can only calculate the top ratio because the user has no way of knowing if a decreasing coverage ratio is due to poor performance of the algorithm or if the system is running out of relevant documents. Basically the user probably does not care – to the user, a declining precision and coverage ratio means that no more relevant topics are being retrieved. Precision and coverage ratio are a measure of user satisfaction because the user would like to see all relevant documents returned per feedback iteration (precision) and cumulatively (coverage ratio). We do not report precision here because of space limitations and the fact that coverage is smoother than precision. However, coverage is also average precision until the ideal search would run out of relevant documents to find and thus indicates the average number of relevant documents returned per screen.

5. A Test Set.

A set of documents labeled by topic can be used to simulate the relevance feedback environment. For a test set we use the Reuters corpus of news articles (www.reseach.att.com/~lewis). The Reuters database is a collection of documents that may be assigned a single topic, multiple topics, or no topic. Eliminating articles with no topics leaves us with 11,367 articles.

Words that are two or less letters in length and the words "and" and "the" have been eliminated. Some articles have multiple topics associated with them so some articles may be relevant for different topics. There are a total of 14,302 topics assigned to the 11,367 articles. After processing there are 27,478 unique words using the Porter (1980) stemmer and 31,478 unique words without stemming.

We define the visibility of a topic as the percent of total documents that have that topic. We will track both coverage ratio as a function of iteration parameterized in the following way: (1) visibility of the topic and (2) the number of documents returned in the preliminary search. The number of returned documents in the preliminary search will be restricted to one or three. Later, we will discuss the issue of no returned documents in the preliminary search. Although we decline to report averages because we believe averages conceal the poor performance of algorithms on low-visibility documents, we do not have the space to report the metrics for all topics (nor would that be especially illuminating) and therefore we report the metrics for selected topics (Table 1). In other words, we will first assume that topic number one is the relevant topic and all others are non-relevant. Then we will assume topic number five is the most relevant document and all others non-relevant, etc.

Examining Table 1, since all "corn" and "soybean" topics are also "grain" topics, all soybean and corn articles will have at least two topics . Thus, when we consider "corn" as the relevant topic, we consider all "grain" articles that are also "corn" articles as relevant while "grain" articles that are not "corn" articles are non-relevant.

Metrics that will be reported are the results of averaging ten experiments with the same initial number of preliminary documents and the same visibility. Each of the ten experiments starts with a different random seed so that the initial set of preliminary documents (picked at random) is different.

Table 1 . Some of the topics in the database:

Rank	Topic name	Number	Visibility
1	earn	3987	33%
5	grain	628	5.5%
10	corn	254	2.2%
15	gnp	120	1.4%
20	soybean	78	.68%
30	iron-steel	67	.58%
40	palm oil	43	.37%
50	fuel	28	.24%
60	lei	17	.15%
70	rapeoil	8	.07%

6. Experimental Results.

In Figure 2, we assume one document retrieved on the initial search for two cases: one case where the topic has a high visibility (33%) and the other case with low visibility (1.4%). For each case, we show the results of two algorithms: SVM using binary features and Rocchio using TF-IDF.

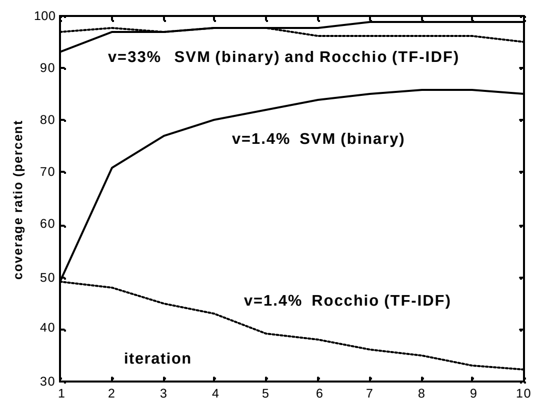

Figure 2. Coverage ratio versus number of iterations for two different visibilities. The dashed line is for Rocchio using TF-IDF features. One document retrieved in preliminary search.

Let us discuss the high visibility case first. Since they have almost identical performance, we have used one label to identify the top two graphs. Recall that precision is identical to the coverage ratio in the first feedback iteration. Both algorithms do very well on the first iteration returning on the average approximately 9.5 relevant documents on a screen of ten documents. At the final iteration, the coverage ratio is approximately 95% indicating that by the tenth iteration an ideal case would return one hundred documents and both these algorithms return cumulatively about 95 documents by the tenth iteration. Therefore both these algorithms do equally well

Now let us examine the case where the visibility is low.

At the first iteration the precision is about 50% indicating, that on the average, five of ten relevant documents would be returned for both these algorithms when one has only one relevant document returned on the preliminary search. However, by the results of the tenth iteration, SVM will have returned approximately eighty-five of the hundred that possibly could be returned in the ideal case while Rocchio will have returned only thirty of hundred relevant documents.

Table 2. Coverage ratio at the tenth iteration using stemmed data. One relevant and nine non-relevant documents assumed returned at the initial search.

Visi-bility	SVM binary	SVM TF	Ite hi TF-IDF	Ide hi TF	Rocchio TF-IDF
33%	99%	100%	95%	95%	95%
5.5%	86%	87%	57%	44%	51%
2.2%	75%	74%	37%	27%	29%
1.4%	85%	85%	34%	32%	32%

Tables 2 and 3 indicate the coverage ratio at the end of the tenth iteration using stemmed and non-stemmed vocabularies, respectively, and assuming one relevant document returned in the initial search. Ide regular is not even reported since its performance is abysmal (zero at the end of ten iterations) probably due to the fact that there are many non-relevant documents used in the initial iteration which forces negative term weights in the query that are then clipped to zero. In Rocchio this effect is reduced because the non-relevant summation term is divided by the number of non-relevant documents while in Ide dec-hi only one non-relevant document is used. Here, the coverage ratio is also average precision at each feedback iteration because the maximum number of documents that could have been retrieved is one hundred and this is less than the actual number of these documents in the collection.

Table 3. Coverage ratio at the tenth iteration using non-stemmed data. One relevant and nine non-relevant documents assumed returned at the initial search.

Visi-bility	SVM binary	SVM TF	Ide hi TF-IDF	Ide hi TF	Rocchio TF-IDF
33%	99%	100%	95%	95%	96%
5.5%	88%	88%	57%	45%	50%
2.2%	74%	75%	38%	28%	29%
1.4%	83%	86%	36%	33%	33%

Examining these two tables, we see that SVM's using either binary or TF features give almost identical performance. Ide dec-hi (with TF-IDF) does slightly better (and statistically so) that the other two Rocchio-type algorithms except for the high visibility case. Based on these tables, SVM is much better. Since there is no difference in SVM's whether one uses binary or TF features, one might as well use binary features as they are easier to obtain and use stemming as this reduces the vocabulary size.

We do not have space to present the tables for three relevant documents returned on the preliminary search but we reach the same conclusions – SVM's are superior and one might as well use binary and stemmed data. The fact that there are three documents returned in the initial search rather than one gives slightly better performance results except for the highest visibility case.

The procedures discussed up to now assumed that there was at least one relevant document returned on the first screen. This is problematic in the case of the low visibility topics. In those cases, one has to go to subsequent screens to find a relevant document. How many screens one has to search will depend on the sophistication of the preliminary query. Our assumption is that one has to examine a number of documents equal to the inverse of the visibility to find one relevant document. In Table 4, we show the coverage ratio after twenty feedback iterations for documents with low visibility. By that time (Table 1), one should have retrieved all documents. The non-SVM algorithms are not shown because they are so poor – approximately zero at the end of twenty iterations.

Table 4. Coverage ratio after twenty feedback iterations using stemmed data. The inverse of the visibility documents are retrieved before one obtains one relevant document.

Visibility	SVM binary	SVM TF
1.4%	86%	86%
.68%	72%	71%
.58%	85%	84%
.37%	100%	95%
.24%	41%	40%
.15%	100%	100%
.07%	100%	94%

The excellent performance can be partially attributed to the facts that the negative examples (of which there are many) may be well separated from the few positive examples. The number of non-relevant documents in the preliminary search is inversely related to the visibility.

7. Conclusions

We have analyzed the performance of SVM-based algorithms and compared them to Rocchio, Ide regular, and Ide dec-hi when the preliminary query gives very few relevant documents. Ide regular performs so poorly that its results are not even reported. When the number of relevant documents in the database is relatively high, all of the algorithms perform well. However, when the visibility is low, SVM using binary features is much better. Obtaining binary features is much simpler than that of obtaining TF features and only requires one pass over the data rather than the two passes required for TF-IDF features. In our experiments we picked a random set of preliminary documents. This has the advantage of allowing us to average over multiple experiments but could be considered to be artificial in that a more realistic front-end would deliver a first set of preliminary documents that are "closer" in some sense. We intend to investigate this case in the future.

Acknowledgements

Thanks go to Vladimir Vapnik for his insights and Thorsten Joachims who supplied the code for the support vector machine optimization problem.

References

Buckley, C., Salton, G., & Allen, J. (1993). Automatic retrieval with locality information using SMART, *Proceedings of the First Text Retrieval Conference (TREC-1)* (pp. 59-72).

Buckley, C., Slaton, G., & Allen, J. (1994). The effect of adding relevance Information in a relevance feedback environment. Proceedings of the Seventeenth Annual International Conference on Research and Development in Information Retrieval.

Drucker, H., Wu, D., & Vapnik, V.N. (1999). Support Vector Machines for Spam Categorization, *IEEE Transactions on Neural Networks*, *10*, 1048-1054.

Dumais, S., Platt, J., Heckerman, D., & Sahami, M. (1998). Inductive learning algorithms and representations for text categorization, *Seventh International Conference on Information and Knowledge Management*. (pp. 148-155).

Harman, D. (1992). Relevance feedback revisited, *Proceedings of the Fifth International SIGIR Conference on Research and Development in Information Retrieval* (pp. 1-10).

Joachims, T. (1997). A probabilistic analysis of the Rocchio algorithm with TFIDF for text categorization, Proceedings *of the Fourteenth International Conference on Machine Learning*, (pp. 143-151). San-Francisco: Morgan Kaufmann.

Joachims, T. (1998), Text categorization with support vector machines: learning with features, *European Conference on Machine Learning* (pp. 137-142).

Korfhage, R.R. (1997). *Information Storage and Retrieval*, Chapter 8, John Wiley, New York.

Lewis, D. (1995). Evaluating and optimizing autonomous text classification systems, *Proceedings of the Eighteenth Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval*, (pp. 246-254)

Lewis, D.D., Schapire, R.E., Callen, J.P., & Papaka, R. (1996). *Training algorithms for linear text classifiers, Proceedings of the Nineteenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* (pp. 298-306).

Mizzaro, S. (1997). Relevance: The whole history, *Journal of the American Society for Information Science*, vol. 48, #9. 810-832.

Porter, M.F. (1980). An algorithm for suffix stripping, *Program*, vol. 14, #3, 130-137.

Robertson, S.E. (1997). The probability ranking principle in IR. *Journal of Documentation*, *34*, 294-304.

Rocchio, J.J. (1971). Relevance feedback in information retrieval, *The SMART Retrieval System: Experiments in Automatic Document Processing*, ed: Gerald Salton, Prentice Hall, 313-323.

Salton, G., & Buckley, C. (1988). Term-weighting approaches in automatic text retrieval, *Information Processing &Management*, *24*, 513-523.

Salton, G., and Buckley, C. (1990). Improving retrieval performance by relevance feedback, *Journal of the American Society for Information Science*, *41*, 288-297.

Saracevic, T. (1975). Relevance: A review of and a framework for the thinking on the notion in Information Science. *Journal of the American Society for Information Science*, 321-343.

Schohn, G., and Cohen, D. (2000). Less is more: active learning with support vector machines, *Proceedings of the Seventeenth International Conference on Machine Learning* (pp 839-846).

Schapire, R.E., Singer, Y., & Singhal, A. (1998). Boosting and Rocchio applied to text filtering, *Proceedings of the Twenty-first Annual International ACM SIGIR Conference on Information Retrieval, SIGIR* (pp 215-223).

Tauge-Sutcliffe, J. (1992). Measuring the informativeness of a retrieval process, *Proceedings of the Fifteen Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* (pp. 23-36).

Tong, S., & Koller, D. (2000). Support vector machine active learning with applications to text classification, *Proceedings of the Seventeenth International Conference on Machine Learning* (pp. 999- 1006). San-Francisco: Morgan Kaufmann.

Vapnik, V. (1982). *Estimation of dependencies based on empirical data*. Springer-Verlag .

Vapnik, V. (1998). *Statistical Learning Theory*, Wiley.