Focused Crawlers vs Accelerated Focused Crawlers

Benjamin Markines Fulya Erdinc Lubomira Stoilova

4/30/2004

1 Abstract

Focused crawlers are an efficient method to build a set of Web pages related to a specific topic. In this project, we explored two different methods to navigate the web given a topic: in the first method, we add all the links from a page to the frontier with a score equal to the relevancy of the parent page; in the second method we add each link to the frontier with a score equal to the relevancy of only the local text around the link on the parent page. We used three types of text classifiers—linear SVM, Naive Bayesian, and cosine similarity—to rate relevancy. The user can specify a topic through a set of relevant documents. From them we selected a set of feature words, via document frequency or mutual information. The classifiers only look for the presence of absence of the feature words. Furthermore, we compared performance over a different number of selected features. The evaluation was inconclusive: the SVM performed slightly better than the other two classifiers in the first method of crawling. Also, the results from using the second method for crawling were not significantly better than the result from the first method. Only the performance of cosine similarity and Naive Bayes was affected by the method of feature selection and the number of features.

2 Previous Work

Traditional focused crawlers maintain their frontier by adding all the links from a fetched page to the frontier with priority equal to the relevancy of the parent. This approach relies on the assumption that pages from a given topic link to other pages in the same topic. While this is a valid assumption in most cases, it is also true that pages contain links to generic sites—such as search engines, IE or Adobe Acrobat home pages—that are not relevant to the user query. Chakrabarti et al. [2] suggests that the crawler can avoid following the irrelevant urls if it determines the relevancy of the link according to the local text surrounding it. He uses the information directly around an HREF in order to predict the topic of the target page. Features are extracted from around a link up to a given distance in the DOM tree.

Chakrabarti uses the terms *baseline* to refer to the traditional focused crawler that navigates through the Web in order to build a database of classified parent and target pages. The *apprentice* is trained on the database and eventually able to guide the crawl by determining the relevancy of new links based on the local context around the HREF. The apprentice is also referred to as an accelerated focused crawler. In this paper we adopted the words *baseline* and *apprentice* with similar meanings. In addition to the Naive Bayes classifier used in Chakrabarti's study, we introduced linear SVM and cosine similarity classifiers. Our work also differs from Chakrabarti's study in the following aspects: we use binomial classifiers instead of multinomial; in the apprentice, we use words as features and he uses word-distance pairs; he studies both online and offline training, while we only used offline training.

Using SVMs for text classification has been studied independently by Joachims [5] and Dumais [4]. Both of them report that linear SVMs outperform significantly all traditional text classifiers and that nonlinear SVMs—higher-degree polynomial and radial basis kernels—give small improvement over linear kernels.

Yang [9] reports that for kNN-like text classifiers, like cosine similarity, document frequency(DF) is more effective than mutual information(MI) for feature selection. Accuracy decreases steadily as the number of selected features is decreased for MI; accuracy increases slightly and then drops dramatically for DF. Chakrabarti [1] suggests that the performance of Naive Bayesian classifiers improves as the number of features increases from 0 to 100 and remains the same in the range from 100 to 500, when using MI. As

for SVMs, Joachims [5] found that performance increases as the number of features increases. It is unclear whether better classifiers guide the crawl better or worse. Pant et al. [6] suggests that sorting the frontier too aggresively might actually hurt the results of the crawl.

3 Implementation

We implemented and tested two types of crawlers against each other. The baseline is a traditional focused crawler. It performs a best-first search: starting from a set of seed pages, it fetches a page, determines its relevancy to the target topic and adds all the links from this page to the frontier with score equal to the relevancy of the parent.

The apprentice is another best-first crawler. It adds new links to the frontier with score equal to the relevancy of the local text around the HREF on the parent page.

We use the open-source crawling engine provided by Gautam Pant [7]. This multi-threaded implementation took care of document retrieval, caching on the local file system, and compliance with the robot exclusion protocol.

Feature selection is done in two ways: using mutual information or document frequency. Mutual information selects the words that appear frequently in the specified topic, but are rare in documents not in the topic. If w is the word and t is the topic, then the mutual information value is defined to be

$$I(w,t) = log\frac{Pr(w,t)}{Pr(t)Pr(c)}$$

$$I(t) = \sum_{i=1}^{m} Pr(c_i)I(t,c_i)$$

Document frequency counts the number of documents in which a word occurs.

We use three types of classifiers for the baseline and the apprentice: cosine similarity, Naive Bayesian, and linear SVMs. Both the Naive Bayesian and the SVM classifiers are binomial: they only take into account whether the feature word appears or not, and disregard the number of times it appears. We use the open-source implementation of linear SVMs provided by Chang et al[3] and transform the output into a probability space using a sigmoid function [4]. For the cosine similarity classifier, we use the standard implementation included with Pant's search engine. We compare the text on the target page to a query containing the 30 feature words that have highest probability for the target topic, as computed for the binomial Naive Bayesian classifier.

The results from the baseline crawls are stored in a mySQL database. A single table contains a set of urls with their topic and parent urls that point to them. During training the apprentice also needs access to the text on the parent, which is stored in a cache maintained by the crawling engine.

To accommodate for the MI feature selection method, we allow the user to train the baseline for several topics at the same time. The user needs to create a separate file with example pages for each target topic. The MI and DF algorithms are normalized for the size of the topics, and the 500 words with the top MI or DF scores over all topics are selected. During testing, we trained for three topics at the same time. We also noticed that, for the purposes of training the SVM, it is useful to keep the number of example pages for each topic similar to each other. The SVM classifier also performs much better if the user includes a forth topic (*Random*) that is described by a set of example pages over a wide range of topics. For example, if the user is interested in topics A, B, and C, she should include example pages for A, B, C, and *Random*. The baseline will perform feature selection over the four topics (*Random* should not generate any features because its example pages are unrelated and thus do not have a defined vocabulary), and then will train one SVM per topic. When training the SVM for topic A, the example pages for topic A will be used as positive instances and the example pages for topics B, C, and *Random* will be used as negative instances. During crawling for topic A, when the SVM encounters a page that is not in topic A, B, or C, it will classify it as *Random*, and thus the page will correctly be classified as *"not in topic A"*. In the absence of *Random*, the SVM is equally likely to assign the random page to topics A, B or C; therefore with probability 1/3 the page will be incorrectly classified as *"in topic A"*. Preliminary results confirmed our intuition that including *Random* strengthens the classifiers' ability to recognize pages irrelevant to the target topic.

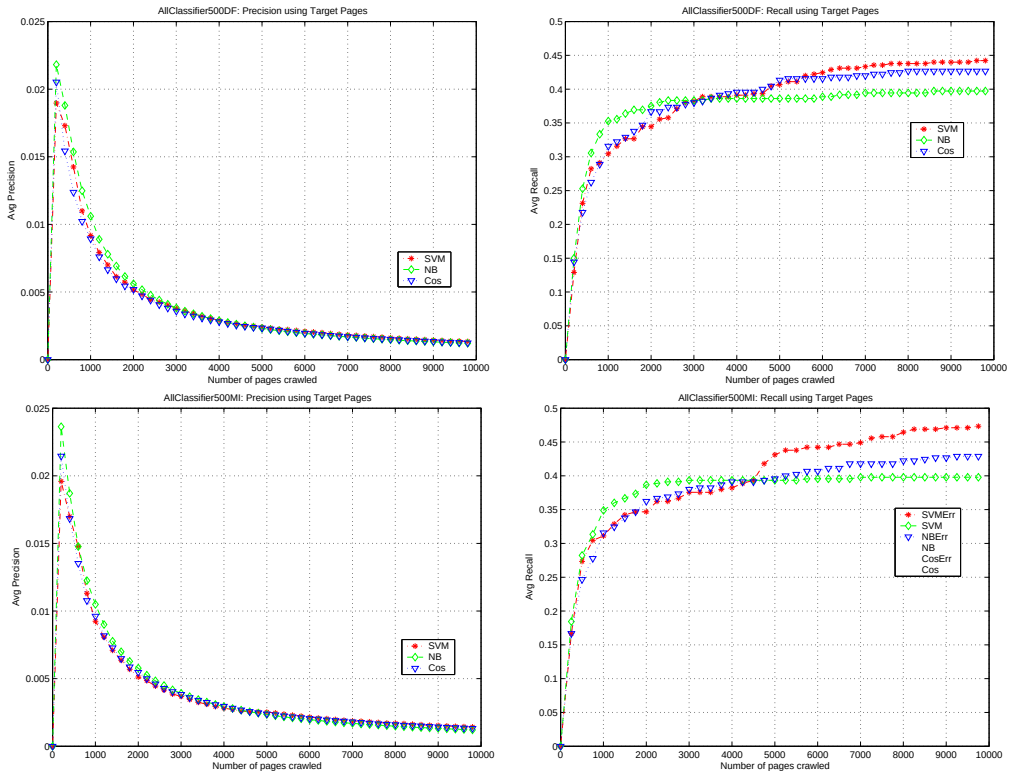

Figure 1: Precision and recall for the baseline crawlers measured by using 30 target pages per topic. The two graphs on the top, show precision and recall when the crawlers used 500 features selected with DF. In the two graphs at the bottom, 500 features are selected using MI. In all the graphs the error bars overlap and the results are not statistically different.

When building the database, we use the baseline to crawl for all three target topics and for the topic *Random*. The apprentice performs feature selection over all topics in the database, picking the words with the top MI or DF scores respectively. We only considered text that appeared around the target links, at distance at most 10 to the left and the right of the HREF in the DOM tree.

Also, during crawling, we do not classify the seed pages provided by the user: we added them to the frontier with the highest possible positive score to ensure that they receive immediate consideration.

4 Results

We used 15 topics of level 3 from Open Directory for evaluation. Topics of level 3 are topics that are at distance 3 from the DMOZ root directory, like Home/Recreation/Aviation/Model_Aviation or Home/Business/Hospitality/Food_Service. The files with example pages for training the baseline include all the urls that Open Directory lists as being part of the target topic or any subtopic of the target topic (for example, Model_Aviation is described by all the urls listed under Home/Recreation/Aviation/Model_Aviation/*).

In order to train the apprentice, we used the same example pages as seeds for the baseline crawlers and crawled for each target topic (as well as *Random*) for 4,000 pages. We performed feature selection for the apprentice over all the entries in the database, looking at text at distance at most 10 from the HREF. However due to time constraints, we only trained the apprentice SVMs using 1,000 instances from each topic.

We used an evaluation framework proposed by Shrinivasan et al. [8]. Recall and precision were measured in two different ways. In the first method, we selected 30 target urls per topic (10 from the topic, 10 from subtopic at depth 1, and 10 from subtopics at depth 2) and we crawled backwards for 2 links to generate

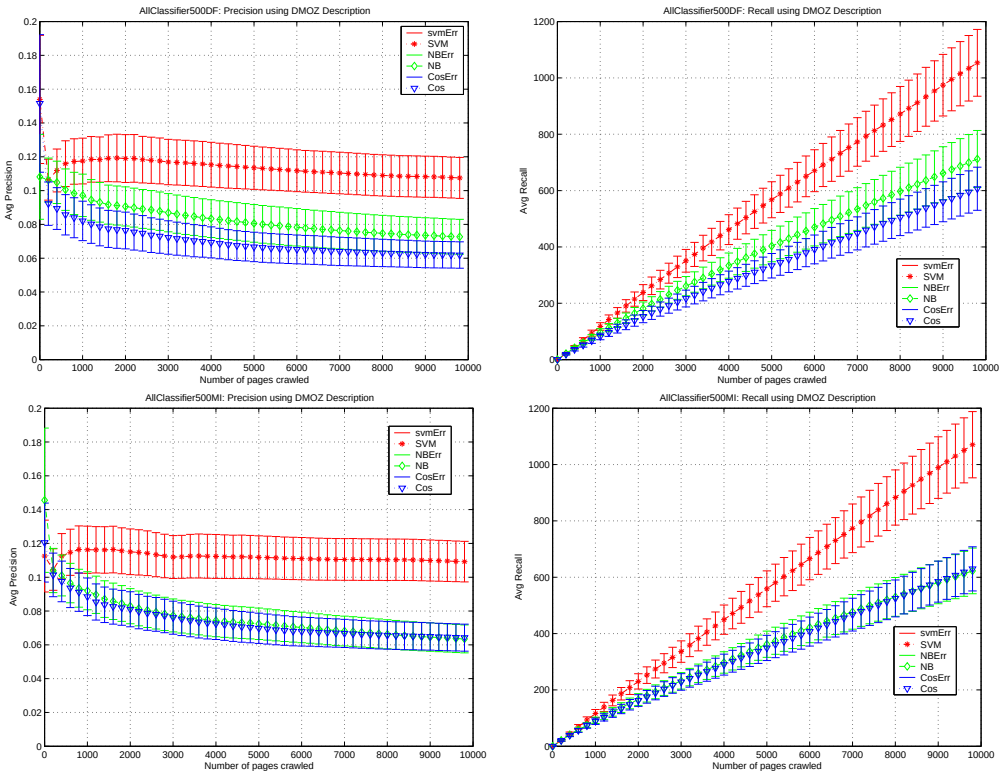

Figure 2: Precision and recall for the baseline crawlers measured using the topic descriptions written by DMOZ editors. The two graphs on the top, show precision and recall when the crawlers used 500 features selected with DF. In the two graphs at the bottom, 500 features are selected using MI.

about 50 seeds per topic. We made sure that all the targets are valid urls and that all of them are reachable from the seeds. We measured recall as the number of targets our crawlers found; we measured precision as the number of targets found over the number of pages crawled.

In the second method, we used the same seeds as in the first method. However this time we looked at all of the pages that our crawlers found and computed cosine similarity between them and the topic descriptions written by Open Directory editors. We used the combined topic descriptions of the urls in the target topic, as well as its subtopics of depth 1 and 2. Recall is as the sum of all the similarity scores of pages found by the crawler; precision is that sum divided by the number of pages crawled.

We ran several types of tests on the baseline. First, we extracted 500 feature words using MI from the DMOZ example pages and crawled it for 10,000 pages starting from 60 seeds that are two backlinks away, as described above. We crawled using the three types of classifiers for all fifteen topics. Then we repeated the same experiment using 500 features extracted with DF. When we measured precision and recall with the first evaluation framework, the results that we got were not statistically different from each other (Fig.1). The performance of any one classifier did not differ from the performance of the other two classifiers and did not depend of the type of feature selection. In the second evaluation framework, the SVM classifier showed better precision and recall than the Naive Bayes and cosine similarity (Fig.2). Naive Bayes performed slightly better than cosine similarity, but it was not statistically significant. The performance of the baseline crawlers did not depend on the type of feature selection. The Naive Bayes performed slightly better when featured were extracted with DF, but it was not significant.

Due to time constraints, we did not test the apprentice extensively. We gathered data to train the apprentice using an SVM baseline crawler with 500 feature words selected with DF. We crawled for 4,000 pages per topic (plus 4,000 pages for *Random*) and created a database with 3 topics (plus *Random*). From the database, we extracted 500 features for the apprentice using DF. We trained the SVMs using only 1,000

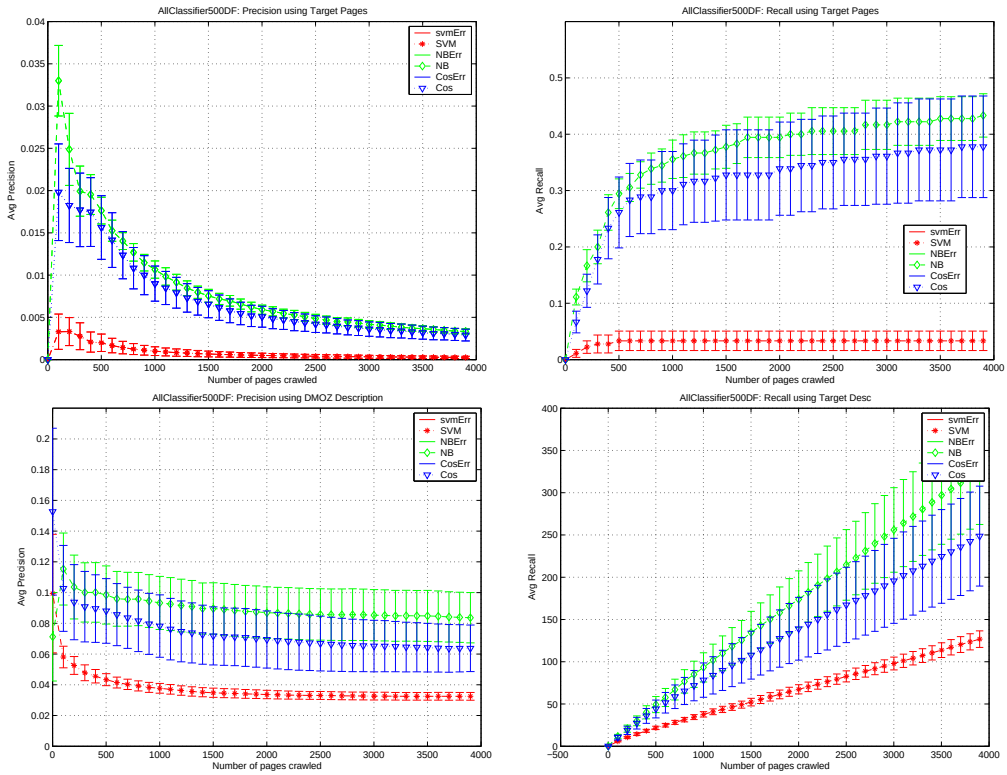

Figure 3: Precision and recall for the apprentice crawlers. In all graphs, data for training the apprentice was gathered by an SVM baseline with 500 feature words selected with DF. The apprentice uses another set of 500 feature words selected with DF from text around HREFs. The two graphs on the top show precision and recall measured using 30 target pages. The two graphs at the bottom show precision and recall measured using the topic descriptions written by DMOZ editors.

instances per class from the database. Finally, we ran the apprentice crawlers for 4,000 pages using the three classifiers. We were only able to test the apprentice on 9 of the topics that we used for the baseline, in 3 sets of three. In both evaluation frameworks, the Naive Bayesian classifier performed the best and the SVM classifier performed the worst. The data for the SVM classifier is statistically different (Fig.3). Our results suggest that Naive Bayes guides the crawl better than cosine similarity, but the data did not show statistically significant difference.

We did not get a significantly better results when using the accelerated apprentice as opposed to using the traditional baseline. The SVM classifier gave very poor results, probably due to the fact that it was not trained on enough instances. According to the first evaluation framework, the precision and recall for the Naive Bayes and the cosine similarity crawlers was the same with the baseline and the apprentice crawlers. In the second framework, precision stayed the same but recall for the apprentice was significantly worse. These results might be due to the fact that we only crawled for 4,000 per topic with the baseline to build a database for the apprentice. That might not be enough data to train the apprentice properly. Alternatively, since many Web pages do not contain a lot of text around their links, it might not be possible to guess the topic of the context from this little information with our types of classifiers.

Finally we were interested in the question of whether aggressive feature selection boosts or hurts the performance of our crawlers. We were only able to test the baseline crawlers. For each of the 15 topics, we selected 100, 300, 500, 700, and 900 feature words using MI and DF and then crawled for 4,000 pages using each classifier. Figure 4 shows recall at 4,000 pages crawled for each classifier. Note that, since precision is defined as recall over the number of pages crawled in both evaluation frameworks, the precision graphs would look the same, only scaled down by a factor of 4,000.

The results in the two evaluation frameworks were contradictory. According to the first evaluation method, for a small number of features, the Naive Bayesian crawler performs significantly better with DF. For DF, performance deteriorates as the number of features is increased past 500. In the second framework, for small number of features, MI is the better feature selection method for Naive Bayes. Performance with MI deteriorates significantly when the number of features is increase past 300.

In the second evaluation framework, performance for both the SVM and the cosine similarity classifier peaks at 500 feature words for MI and DF. The performance of the crawlers is not significantly affected by the method of feature selection.

In the first evaluation framework, the performance of the SVM classifier is not statistically different over different method for feature selection of number of features. For small number of features, the cosine-similarity classifier performs better with DF; for a larger number of features, the method of feature selection does not matter.

These results suggest that the number of features and feature selection method affect the Naive Bayesian and cosine-similarity crawlers more than it affects the SVM crawlers and that a Naive Bayesian classifier might benefit from a smaller number of features.

5 Future Work

The framework of the system allows for expansion to other types of classifiers. We would have liked to implement and test a classifier based on Kanerva's distributed memory model and test the Circle classifier developed by Arijit Sengupta. We did not have time to test extensively linear SVMs for the apprentice and we would like to complete these experiments, as well as look at one-class SVMs that seem more suitable for crawling because they only require positive instances for training and do not need examples of pages not in the topic. Finally, this study tries to predict the relevancy of a url only based on text in the parent Web page. It would be interesting to look at accelerated focused crawlers that take into account the spatial location of links or some other non-semantic information.

References

[1] Soumen Chakrabarti, Byron Dom, Rakesh Agrawal, and Prabhakar Raghavan. Scalable feature selection, classification and signiture generation for organizing large text databases into hierarchical topic taxonomies. *The VLDB Journal*, 7:163–178, 1998.

[2] Soumen Chakrabarti, Kunal Punera, and Malella Subramanyam. Accelerated focused crawling through online relevance feedback. In *WWW, Hawaii*. ACM, May 2002.

[3] Chih-Chung Chang and Chih-Jen Lin. Libsvm – a library for support vector machines. 2001.

[4] Susan Dumais. Using svms for text categorization. *IEEE Intelligent Systems*, July/August 1998.

[5] Thorsten Joachims. Text categorization with support vector machines: Learning with many relevant features. LS-8 Report 23, University of Dortmund, Computer Science Department, 1998.

[6] Gautam Pant, Padmini Srinivasan, and Filippo Menczer. Exploration versus exploitation in topic driven crawlers. In *WWW Workshop on Web Dynamics*, 2002.

[7] Gautam Pant, Padmini Srinivasan, and Filippo Menczer. Crawling the web. *In M. Levene and A. Poulovassilis, eds.: Web Dynamics*, 2004.

[8] Padmini Srinivasan, Filippo Menczer, and Gautam Pant. A general evaluation framework for topical crawlers. Final version to appear in Information Retrieval.

[9] Yiming Yang and Jan 0. Pedersen. A comparative study on feature selection in text categorization. *Proceedings of the Fourteenth International Conference on Machine Learning*, 1997.

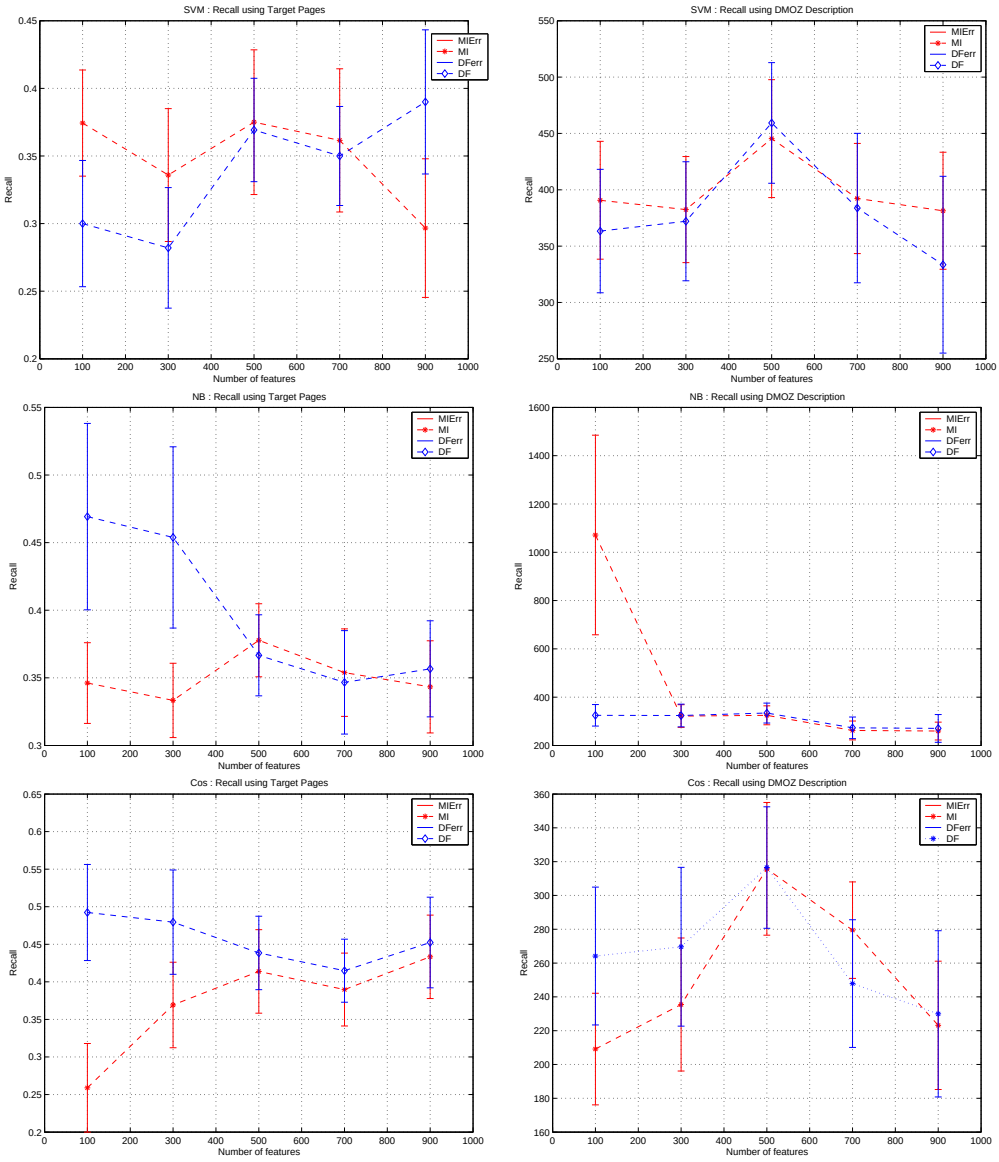

Figure 4: Recall for the baseline crawlers at 4,000 crawled pages for different number of selected features. The two graphs on the top show recall for the SVM crawler: using 30 target urls in the first graph and DMOZ topic descriptions in the second graph. The graphs in the middle refer to Naive Bayes, and the graphs at the bottom are for cosine similarity. We do not show precision graphs because precision here will be just recall divided by 4,000 and thus the graphs look the same.