Using Support Vector Machines Classifier to Improve the Performance of Reinforcement Learning based Web Crawlers

Hamid Reza Motahari Nezhad
motahary@aut.ac.ir

Ahmad Abdollahzadeh Barfourosh
ahmad@ce.aut.ac.ir

Computer Eng. & IT Faculty, Amirkabir University of Technology
Tehran 15914, Iran

ABSTRACT
The main contribution of this paper is introducing an approach for expanding the crawling methods of Cora spider, as a RL-based spider. We have introduced novel methods for calculating the Q-Value in reinforcement learning module of the spider. The proposed crawlers can find the target pages faster and earn more rewards over the crawl than Cora's crawlers. We have used Support Vector Machines (SVMs) classifier for the first time as a text learner in Web crawlers and compared the results with crawlers which use Naïve Bayes (NB) classifier for this purpose. The results show that crawlers using SVMs outperform crawlers which use NB in the first half of crawling a Web site and find the target pages more quickly. The test bed for the evaluation of our approaches was Web sites of four computer science departments of four universities, which have been made available offline.

Keywords: Web Crawling, Focused Crawling, Reinforcement Learning, Text Classification.

1. INTRODUCTION

Focused crawling [2, 3, 4] s a method, which is able to crawl particular topical portions of the World Wide Web quickly and efficiently by following only the most interesting links without any need to explore the entire Web [2]. Many researches [2,3,4,5,6,7,8,9,10,11,12,13,14,15] have been done to improve the functionality and effectiveness of focused crawlers. The major problem of focused crawling is how to find the path for the crawler, towards relevant pages to the focused topic(s). The most important feature of reinforcement learning which makes it appropriate for focused crawling of the Web is its ability to model the future reward of an action in form of delayed rewards. So, it can give an estimation of the amount of rewards (target pages) which an agent is able to earn in the future by following a hyperlink.
Previous researches in Cora search engine [4, 5, 6] show that focused crawlers of the Web, which use reinforcement learning (RL) as their learning method have achieved a better performance and are more efficient than other focused crawlers. RL-based Crawler of Cora search engine uses Naïve Bayes (NB) classifier for learning the text in the neighborhood of hyperlinks. The result of the classification of neighborhood text of hyperlink is used to estimate Q-value of each hyperlink. Support Vector Machines (SVMs) classifier outperforms Naïve Bayes classifier in text classification. In this paper, we have used both classifications (NB and SVMs) in a RL-based crawler and examined them against different parameters, which affect on the performance of RL-based crawler. The results show that crawlers using SVMs outperform crawlers which use NB in the first half of crawling a Web site and find the target pages more quickly. The test bed for evaluation of our approaches was Web sites of four computer science departments of four universities.
In section 2, we give an overview of related works. Section 3 describes how to crawl the Web using reinforcement learning and what is our extension to this approach. Section 4 explains the design and implementation of the crawlers. Section 5 represents implementation results and we finally conclude and discuss future works in section 6.

2. RELATED WORKS

Reinforcement learning has been used in two kinds of Web search applications. Firstly, it was used for navigation of users in a recommender system, WebWatcher [18]. This system learns a Q-function for each word in pages and recommends the hyperlinks, which maximize the sum of the corresponding Q-functions. The second approach to focused crawling uses reinforcement learning (RL) to spider the Web efficiently, in Cora search engine [4,5,6], is three times more efficient than Breath-first crawler, which is commonly used by general purpose search engines, and also is more efficient than other focused crawling methods [5]. This paper is an extension to approaches of Cora in crawling of the Web as it has been described in the next sections.

3. CRAWLING THE WEB USING REINFORCEMENT LEARNING

Reinforcement learning [19], as a type of semi-supervised leanings, is a framework for learning from rewards and punishment. This method is appropriate for learning by trial and error from dynamic environments and situations with delayed rewards. The specific reasons that make reinforcement learning approach appropriate for Web crawling is [5]: (1) Web crawling provides situations with delayed rewards (rewards which are attainable by following a hyperlink several hops farther). (2) We can calculate the performance as the sum of discounted rewards achieved by the crawler. (3) Dynamic environment (Web sites), which an on-line RL-based crawler can learn new models and adapts itself to the changes in the environment. In following subsections, we will discuss reinforcement learning and how focused crawling problem can be mapped onto it and our extensions in this regard.

Using Reinforcement Learning for Web Crawling

In this section, we describe that how we map our problem, crawling of the Web, to reinforcement learning framework. In Web crawling, related pages to the focused topic are immediate rewards, following a hyperlink is considered as an action, and state is the set of: (1) all of hyperlinks have been visited up to now and, (2) all of related pages on the Web to be visited [5]. The action space is very large and dynamic since it depends on what pages on the Web have been visited yet. The state space is also very large and difficult to generate. In practice, we make some simplifying assumption in order to make the problem manageable and to aid generalization [5]. Researchers of Cora [5] has made following simplifying assumption that we will also used them: (1) we consider that state is independent to which related pages has been visited and remained to be visited. Considering visited hyperlinks are actions, using this assumption, all of states are mapped into one state and our problem changes to selecting the best available action in any given time. (2) We assume that we can obtain an estimation of the amount of future rewards of following a hyperlink by a mapping from its neighborhood text to a scalar value. This mapping is done by solving this regression problem through classification [20]. Cora search engine used Naïve Bayes classifier to classification of neighborhood text of hyperlinks. Support vector machine classifier [21], which is a based on computational learning theory [22], has achieved a better performance in text classification than naïve Bayes [23]. One of the extensions that we have applied is using this text classifier and comparing the results with the result of using naïve Bayes.

Naïve Bayes Classifier: In this model of learning, it is assumed that texts are generated from a parametric model and training data collection is used to estimate these parameters [25]. In testing phase, for each new document, using learned parameters and Bayes rule, the probability of generation of this document by each class is computed. Classification is selection of the class with the highest probability for words of this document. This model of learning uses Bayes rule and assumes that occurrence of each word in a document is independent of occurrence of other words in class of that document and the place of its occurrence, and so called naïve Bayes.

Support Vector Machines Classifier (SVMs): This model of learning is based on "Structural Risk Minimization" principle in computational learning [22]. The basic idea of this principle is to find hypothesis *h* for which we can guarantee the lowest true error [23]. The true error gives the probability of the error of the model for an unseen and randomly selected test example. To achieve this goal, this model finds hyperplane with the maximum margin of separation to classes of training data. Transductive Support Vector Machines (TSVMs) is a kind of SVMs that was introduced by Joachimes [24]. The aim of this classifier is to learn from a small number of training data (as training and testing data) in training phase. TSVMs has outperformed SVMs in text classification [24]. While SVMs uses an inductive inference to find a decision function with low

rate of errors in all distributions of examples (as train and test), TSVMs uses transductive inference to classify a set of examples (training data) with the lowest error. Some of the characteristics of text classification problems which make TSVMs, a very good text classifier are: high dimensionality of input space (collections with more than 10,000 features), sparseness of vectors of input space, and low number of irrelevant features [23,24].

4. DESIGN AND IMPLEMENTATION

Design of the RL-based Crawler

The overall architecture of the system is illustrated in figure 1.

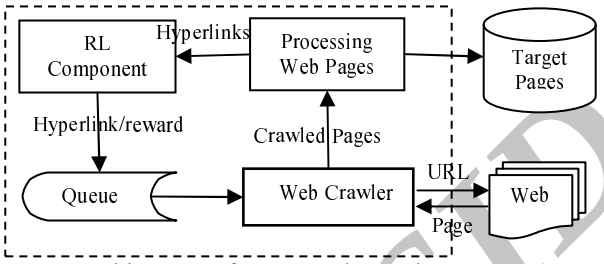


Figure 1. Architecture of RL-Based Crawler

We study the system in two phases: learning and testing. Figure 1 shows the infrastructure of both phases in which the RL component (and the text classifier in it) in learning phase is trained using training Web sites and in test phase it estimates the amount of reward of each hyperlink. Web crawler is a program, which fetches the URL with the highest estimated reward from the prioritized queue and downloads it from the Web. Web page collections are downloaded in a phase for preparation of test bed and made available offline in a database using a hash mechanism for retrieval of a requested Web page using the corresponding URL.

Learning Phase: In this phase, reinforcement learning based agent aims to learn a generalized rule for computing Q-value of a hyperlink using the neighborhood text of hyperlink. This rule helps the agent to estimate the amount of future reward of each hyperlink based on its neighborhood text in the test phase.

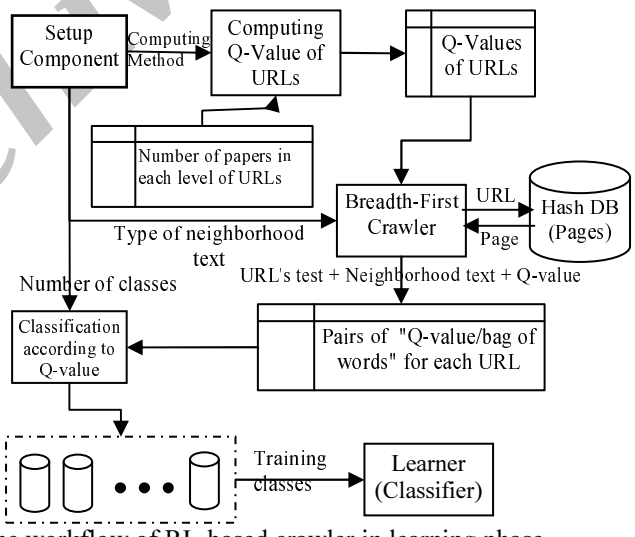


Figure 2. The workflow of RL-based crawler in learning phase

Figure 2 depicts that, a Q-value for each URL is calculated according to the number of papers which are available at different levels of the URLs and based on a specific function. The formal specifications of functions that are used in calculation of Q-values are given in figure 3. These functions, except Cutoff, are also evaluated in Cora. We studied their behavior using SVMs classifier in comparison to NB. After the calculation of the Q-values, URL's text and its neighborhood text are extracted and the pair of "Q-

value/associated text" for each URL is stored in a list. We have used three types of neighborhood text: near text (100 words before, and after the URLs), relative text (near text + previous heads and subheads of the text in page, which contains the URL) and full (whole text of the page, which contains the URL). The number of classes is considered between 3 to 5. The number of classes is limited to 5, since the accuracy of naïve Bayes classifier for hypertexts and their corresponding texts decreases to around 50% for 6 classes. Given pairs of "Q-value\bag of words" classified to some predefined classes, they are used as training data of a classifier. This classifier is used as the input of the testing phase.

Testing Phase: In this phase, Q-values of the URLs are estimated using their neighborhood text. For each page, which the crawler finds, it extracts its URLs and its corresponding neighborhood text. For each URL, these texts are represented to the trained classifier and it returns the probability of pertaining the text to documents in each class. These probabilities are used to calculate the Q-value of the URLs as follows [5]:

$Q(text) = \sum_{i=1}^{n} P_{c_i}(text) * Q\_Avg(C_i)$, which n indicates the number of classes, $Pc_i(text)$ is the probability of pertaining the text to class number i, and $Q\_Avg(c_i)$ is the average of Q-values of texts in class i. The workflow of system in the testing phase is illustrated in figure 4.

Distance:
Q-value(URL)= γ ^ (distance to the nearest reward)
Future(Three):
Q-value(URL) = 1 for immediate reward, γ for future, zero for none.
Future(Four):
Q-value(URL) = 1 for immediate reward, γ for one-step, γ ^2 for two-steps, zero for none.
Future(Five):
Q-value(URL) = 1 for immediate reward, γ for one-step, γ ^2 for two-steps, γ ^3 for three-steps, zero for none.
Future(Parallel):
Q-value(URL) = ∑Num(reward) * (γ ^ distance)
Immediate:
Q-value(URL) = 1 If the link is a paper else 0.
Cutoff:
Calculates according to path, if value < $cutoff, gives value of 0.

```
for (my i=0; i < 10; i++)
{ # Bonus reward for each item at this level
  for (my j=0; j < depth[i]; j++)
  { score_cutt += γ ** count;  count++}
   # link to move to next level
  count++;}
score_cutt = 0 if (score_cutt < cutoff);
```

Figure 3. The formal specification of functions, which are used for calculation of Q-values in learning phase

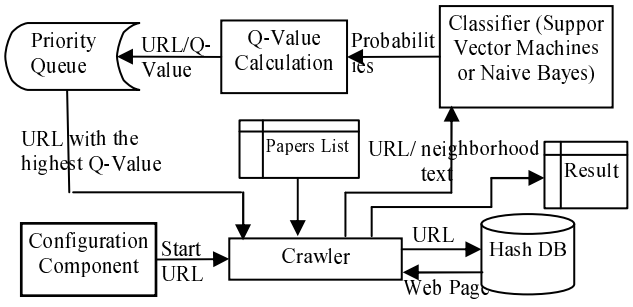


Figure 4. The workflow of RL-based crawler in testing phase

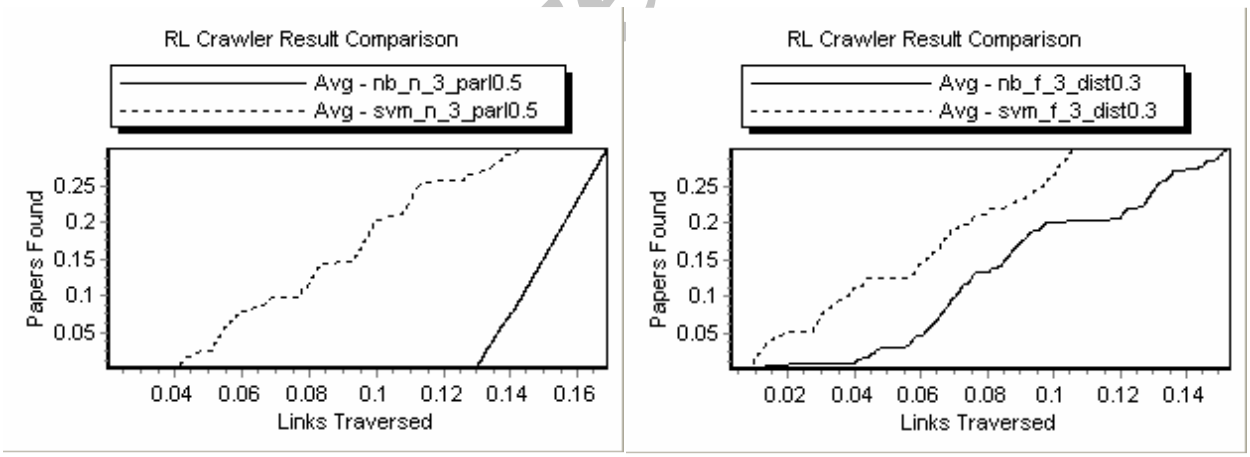
Experimental Setup

The system is designed to work on-line on the Web, but we trained and tested it offline, like Cora spider [1]. In October 2002, we crawled Web sites of four computer science departments in Boston, Brown, Pittsburg and University of California at Davis (UCDavis). There were 23044 HTML pages, 129758 hyperlinks and 4169 paper files (target pages). For each run, we used three of them as the training set and the other as the test Web site for the evaluation of the performance of crawling methods. The reported results for each crawling method are averaged over the performance of that crawling method for the four Web sites, separately.

5. RESULTS

A crawler based on reinforcement learning is considered to be more efficient than other RL-based crawlers, since it is able to gain a maximum number of rewards, while traversing a fewer number of hyperlinks. We evaluated the crawling methods against each other based on two parameters: (1) the ratio of the number of rewards (target pages) to the number of traversed hyperlinks. We visualize this parameter by drawing the number of crawled target page against the number of traversed hyperlinks in a chart and comparing them. (2) the amount of rewards that each crawler achieves during a crawling session. The goal of a RL-based crawler is to maximize achieved rewards during crawling. This reward is computed as the integral of the space under the curve in a chart, which its vertical axis is the number of crawled target pages and its horizontal axis is the number of traversed hyperlinks.

Using support vector machines classifier in comparison to naïve Bayes improved the performance of crawlers especially in the first half of the crawling. Crawlers which use SVMs find the target pages faster than other approaches, which show the ability to find their path towards target pages at the first part of a crawling session. Figures 5.a and 5.b depict a comparison of crawlers, which use SVMs and NB. In this figure, *Avg* in the initialization of the name of methods implicates that the results are averaged over the results of crawling four Websites of test bed. The overall form of the name of methods is: Classifier_NeighborhoodText_ClassNum*FunctiunGamma,* which the values of *Classifier* is SVMs or NB*, NeighborhoodText* can be *n* (near), *r* (related), or *f* (full). *ClassNum* is 2, 3, 4 or 5. *Function* is the function name for calculating Q-Values, and finally *Gamma* is considered to be 0.1, 0.3 or 0.5.



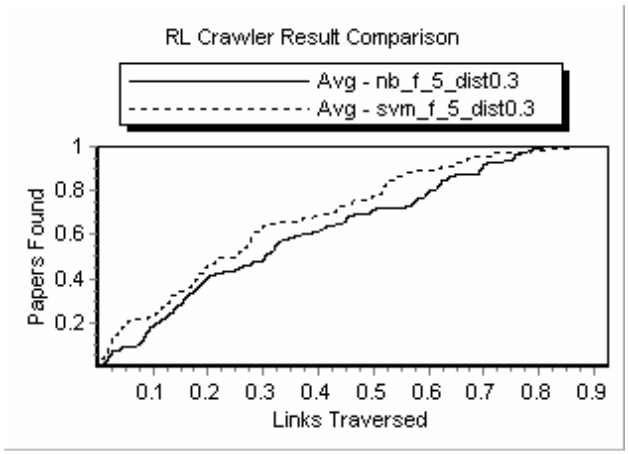
5.a- Comparison of two methods which use *future(parallel)* function (near text, 3 classes and gamma = 0.5) for calculating Q-values

5.b- Comparison of two methods which use *distance* function (full text, 3 classes and gamma = 0.3) for calculating Q-values

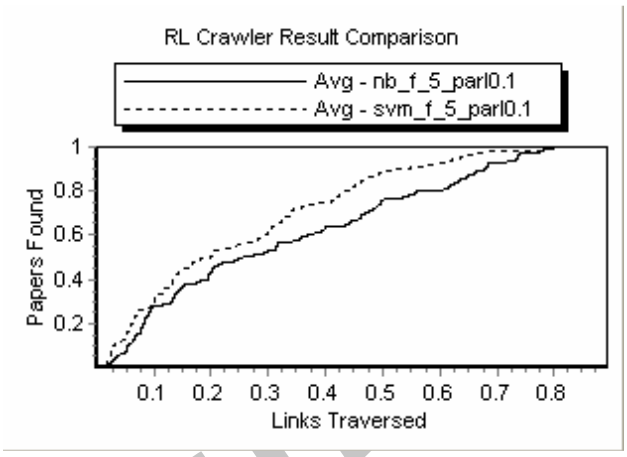
Figure 5. Comparison of RL-based spiders in using SVMs and NB classifiers in the first 30 percent of crawling session.

As figures 5.a and 5.b show the crawlers which use SVMs classifier outperform crawlers which use NB classifier. We selected the first 30 percent of the crawling to provide a better view of crawler's performance in the first steps of crawling and moving towards target pages from the start. Figures 6.a and 6.b show the comparison of two crawlers in the whole crawling session for finding computer science papers in computer

science departments. The figures 6 indicate that the crawlers which use SVMs outperform crawlers which use NB classifier for whole crawl session.



6.a- Comparison of two methods, which use *distance* function for calculating Q-values



6.b- Comparison of two methods, which use *future (parallel)* function for calculating Q-values

Figure 6. Comparison of reinforcement learning spiders using SVMs and NB classifier in whole crawl session.

We have implemented a variety of functions for calculating and their comparison through the first evaluation parameter is not practical. The second evaluation parameter which is the amount of achieved reward is used for comparing different crawlers with each other. Table 1 shows comparison between the averaged amount of rewards which different focused crawlers achieve for the first half of the crawl (50%) and whole of it (100%).

Table 1. Comparison of the amount of reward which have been achieved by different methods of RL-based crawling

Method (50%)	Rewards	Method (100%)	Rewards
svm_n_4_cut_g0.3	88.481055	nb_n_4_cut_g0.3	81.815967
Nb_n_4_cut_g0.3	88.36996	nb_r_3_cut_g0.5	78.335362
svm_n_5_cut_g0.5	87.569967	nb_r_3_parl0.1	78.04647
svm_r_5_cut_g0.5	87.370997	nb_r_4_dist0.5	77.968872
svm_r_5_five0.5	87.259045	nb_r_3_parl0.3	77.94723
svm_r_3_parl0.3	87.02282	nb_r_4_dist0.3	77.87946
Nb_n_3_cut_g0.5	86.977685	nb_n_3_cut_g0.5	77.364047
Nb_r_4_four0.5	86.683362	nb_r_3_dist0.3	77.352802
svm_r_4_cut_g0.5	86.627875	nb_r_3_cut_g0.3	77.25757

As the results show, crawlers which use SVMs are able to acquire more reward in the first half of the crawl. But, considering the whole crawling session (100%), crawlers which use NB classifier achieve more rewards, since most of them act better in the second half of the crawling and their overall reward is greater than other crawlers. This observation is shown in table 1. Considering this observation, we designed and implemented a crawler which uses the positive features of both crawlers, i.e., it uses SVMs classifier in the first 30% of a crawling session and then continues with the NB classifier. The experimental result shows that crawlers using this approach achieve a better performance when using both of the evaluation parameters (ration of target pages found to hyperlinks traversed, and the amount of achieved reward). We called this approach the *ChangePolicy*.

We have compared effect of the *ChangePolicy* approach in crawlers on one focused crawler, and a breadth-first crawler. Figures 7.a and 7.b show the comparison between two crawling methods (one based on

ChangePolicy (and SVMs), the other using NB classifier, a breadth-first search crawler, and a focused crawling method which does not model future reward of a hyperlink (immediate function).

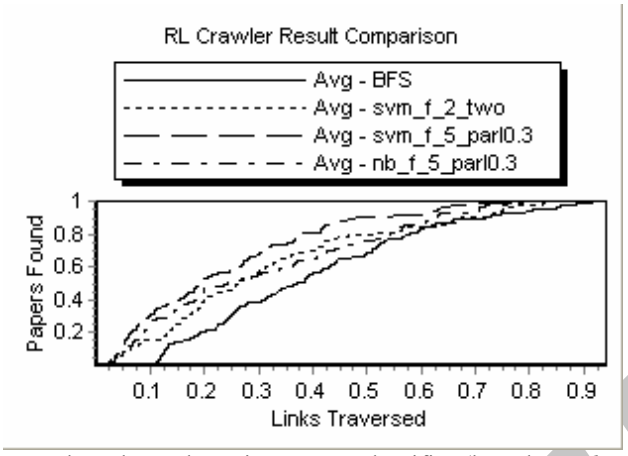


Figure 7. Comparison of a RL-based crawler using SVMs classifier (based on *ChangePolicy* approach) with same method using NB, BFS and immediate focused crawler

As it is shown in figures 7, focused crawlers which use SVMs classifier in the *ChangePolicy* approach outperform RL-based crawlers which use NB, as well as, focused crawlers which do not model future rewards (uses immediate function for Q-value calculation, here svm_f_2_two) and also breadth-first crawler which is used in most of the general purpose search engines.

6. CONCLUSION AND FUTURE WORKS

In this paper, we introduced some novel approaches for expanding reinforcement learning based crawling of the Web in Cora. We used support vector machines for the first time as a text classifier in focused crawlers and the results shows that this crawlers achieve better effectiveness in the first half of crawling and in finding the path towards target pages at the first part of the crawling session. We proposed and implemented a *ChangePolicy* approach, in which crawler changes its policy for calculating Q-values during a crawling session.

One of the potential future works is using companies as a test bed which has a single target page and has been used in Cora, since this test bed is able to show the intelligence of a RL-based crawler and its ability to find its path towards target page better than the currently used test bed. In the *ChangePolicy* approach, policy changes manually. The other area for future work is to monitor the performance of the crawler and change the policy when the behavior of the crawler changes significantly. Reinforcement learning has shown a good ability to learn the model of an environment. As a potential future work, it would provide us with a greater performance, if we use reinforcement learning to learn the structure of Web sites as well as their contents.

7. REFERENCES

[1] H. R. Motahari Nezhad, A. A. Barfourosh**,** *Expanding Reinforcement Learning Approaches for Efficient Crawling of the Web*, The World Multi Conference on Systematics and Cybernetics and Informatics (SCI'2003), July 27 - 30, 2003, Orlando, Florida, USA.

[2] Chakrabarti S., Van Der Berg M., and Dom B., *Focused crawling: a new approach to topic-specific Web resource discovery*, In Proceedings of the 8th International World-Wide Web Conference (WWW8), 1999.

[3] Chakrabarti S., Van Der Berg M. and Byron E. Dom, *Distributed Hypertext Resource Discovery Through Examples*, Proceedings of the 25th VLDB Conference, Edinburgh, Scotland, 1999.

[4] McCallum A. K., Nigam K., Rennie J. and Seymore K., *Automating the construction of internet portals with machine learning*, In Information Retrieval, 1999.

[5] Rennie J. and McCallum A., *Using reinforcement learning to spider the web efficiently*, In Proceedings International Conference on Machine Learning (ICML), 1999.

[6] McCallum A. and Nigam K., Rennie J. and Seymore K., *Building domain-specific search engines with machine learninig techniques,* In AAAI-99 Spring Symposium on Intelligent Agents in Cyberspace, 1999.

[7] Diligenti M., Coetzee F., Lawrence S., Giles C. L. and Gori M., *Focused Crawling using Context Graphs*, 26th International Conference on Very Large Databases, VLDB 2000, Cairo, Egypt, pp. 527–534, 2000.
[8] Raghavan S. and Garcia-Molina H., *Crawling the Hidden Web*, Stanford Digital Libraries Technical Report, 2000.
[9] Mukherjea S., *WTMS: A System for Collecting and Analysing Topic-Specific Web Information*, In Proceedings of the 9th International World Wide Web Conference, Amsterdam, Netherlands, May 15-19, 2000.
[10] Menczer F., Pant G., Srinivasan P. and Ruiz M., *Evaluating Topic-Driven Web Crawlers*, In Proceedings of the 24th Annual International ACM/SIGIR Conference, New Orleans, USA, 2001.
[11] Aggarwal C., Al-Garawi F. and Yu. P., *Intelligent Crawling on the World Wide Web with Arbitrary Predicates*, In Proceedings of the 10th International World Wide Web Conference, Hong Kong, May 2001.
[12] Najork M., and Wiener J., *Breadth-First Search Crawling Yields High-Quality Pages*, In Proceedings of the 10th International World Wide Web Conference, Hong Kong, May 2001.
[13] Chakrabarti S., Jaju R., Joshi M. and Punera K., *Analysing fine-granied hypertext features for enhanced crawling and topic distillation*, Bulletin of the IEEE Computer Society Technical Committee on Data Engineering, 2002.
[14] Chakrabarti S., Punera K. and Subramanyam M., *Accelerated Focused Crawling through Online Relevance Feedback*, In 12th World Wide Web Conference, Hawaii, may 2002.
[15] Ehrig M., *Ontology Focused Crawling of Documents and Relational Metadata*, Master Thesis, Institut fÄur Angewandte Informatik und Formale Beschreibungsverfahren UniversitÄat Karlsruhe (TH), Germany, January 31, 2002.
[16] De Bra P., Houben G., Kornatzky Y. and Post R., *Information Retrieval in Distributed Hypertexts*, In Proceedings of the 4th RIAO Conference, 481 - 491, New York, 1994.
[17] Kleinberg J., *Authoritative sources in a hyperlinked environment*, Proceedings of 9th ACM-SIAM Symposium on Discrete Algorithms, 1998. Extended version in Journal of the ACM 46(1999). Also appears as IBM Research Report RJ 10076, May 1997.
[18] T. Joachimes, D. Fritag, and T. Mitchel, WebWatcher: A tour guide for the World Wide Web, In Proceedings of IJCAI-97, 1997.
[19] Sutton R. S., Barto A. G., *Reinformcement Learning: An Introduction*, MIT Press, 1998.
[20] Torgo L. and Gama J., *Regression using classification algorithms*, Intelligent Data Analysis, 1(4), 1997.
[21] Cortes C., Vapnik V., *Support-vector networks*, Machine Learning, 20:273-297, November 1995.
[22] Vapnik Vladimir N., *The Nature of Statistical Learning Theory*, Springer, New York, 1995.
[23] Joachims T., *Text Categorization with Support Vector Machines: Learning with Many Relevant Features*, Proceedings of the European Conference on Machine Learning (ECML), Springer, 1998.
[24] Joachims T., *Transductive Inference for Text Classification using Support Vector Machines*, Proceedings of the International Conference on Machine Learning (ICML), 1999.
[25] Lewis David D., *Naive (Bayes) at forty: The independence assumption in information retrieval*. In ECML-98, 1998.