Improved Focused Web Crawling Based on Incremental Q-Learning

Yunming Ye[1], Yan Li[1], Joshua Huang[2], and Xiaofei Xu[1]

[1] Shenzhen Graduate School, Harbin Institute of Technology, Shenzhen 518055, China
yym_sjtu@yahoo.com.cn
[2] E-Business Technology Institute, The University of Hong Kong, Hong Kong
jhuang@eti.hku.hk

Abstract. This paper presents *IQ-Learning*, a new focused crawling algorithm based on incremental Q-learning. The intuition is to extend previous reinforcement learning based focused crawler with the incremental learning mechanisms so that the system can start with a few initial samples and learns incrementally from the knowledge discovered online. First, a sample detector is used to distill new samples from the crawled Web pages, upon which the page relevance estimator can learn an updated estimation model. Secondly, the updated page relevance information is fed back to the Q value estimator constantly when new pages are crawled so that the Q value estimation model will be improved over time. In this way, the reinforcement learning in focused crawling becomes an incremental process and can be more self-adaptive to tackle the complex Web crawling environments. Comparison experiments have been carried out between the *IQ-Learning* algorithm and other two state-of-the-art focused crawling algorithms. The experimental results show that *IQ-Learning* achieves better performance in most of the target topics.

Keywords: Focused Crawler; Q-Learning; Incremental Learning; Web

1 Introduction

A Web crawler is an information gathering system that traverses the Web by following the hyperlinks from page to page and downloads Web pages that are interested. General Web crawlers visit the Web in an unselective mode. They aim at collecting Web pages as many as possible to build search engines. Different from the general crawler, a focused crawler [1] is an intelligent Web crawler that traverses the Web selectively to download pages in some predefined target topics. Given a search topic and a predefined maximum download number, the goal of a focused crawler is to collect as many relevant pages as possible while retrieving as fewer irrelevant pages as possible in the crawling process.

Focused crawlers are very useful in several Web applications [2], such as collecting Web pages with specific topics for domain-specific search engines, archiving specific page collections for a digital library, and gathering systematic information in some specific topics for market research or survey of literature on the

© A. Gelbukh (Editor)
Advances in Natural Language Processing
Research in Computing Science 18, 2006, pp. 139-150

Web for a scientific research. Therefore, it has attracted much attention in recent years. Several useful methods for building focused crawlers have been proposed, such as the PageRank value based crawling strategy [3], reinforcement learning [4], the evolutionary multi-agent system [5], the statistical modeling method [6], the 'critic-apprentice' learning paradigm [7], to name a few.

Among previous works, reinforcement learning has been testified to be a very effective method for focused crawling [4, 7]. From an agent perspective, a focused crawler is an intelligent agent that interacts with the environment (the Web) and attains goal states (picks up relevant pages). It's natural to fit the crawling process into a reinforcement learning framework: the crawling process is to search an optimal decision making strategy which can be modeled as a discounted function of state, action and reward, where the number of relevant pages collected is the state of the crawler, following a hyperlink is an action, and finding a relevant page is an immediate reward. In [4], Rennie and et al. gave a first solution to design a focused crawler based on reinforcement learning, and their experiments showed good results. However, their method has two limitations:

1) the reward function is learned offline from manually labeled sample pages, which is too time-consuming as the training set may contain thousands of pages that collecting and labeling of them is costly;

2) as the action-reward model is built by offline learning and the model stays unchanged during crawling, the static learned models cannot quickly adapt to the diverse Web environment where the content of Web pages as well as the link structure for different topics are very diverse, and different target topics may require different crawling strategies. The essence of these problems lies in the lack of incremental learning capabilities.

In this paper, we present a new focused crawling algorithm *IQ-Learning*, to enable the focused crawler to learn incrementally from online crawling information so that the reward-action model can be adapted and improved over time. In its incremental *Q-learning* learning framework, the crawler system learns incrementally from the knowledge discovered online through two mechanisms. First, a sample detector is used to distill new page samples from the crawled pages, upon which the page relevance estimator can learn an updated estimation model. Secondly, the updated page relevance information is fed back to the Q value estimator constantly when new pages are crawled so that the Q value estimation model will be improved over time. With incremental learning, the algorithm can start from a few initial samples and be self-adaptive to different crawling environments. Moreover, the reinforcement learning framework also endows it with the ability to make tradeoff between exploiting immediate rewards and exploring future rewards efficiently.

We have compared *IQ-Learning* algorithm to other state-of-the-art focused crawling algorithms. Experimental results showed that *IQ-Learning* got better performance in English topics as well as Chinese topics. It's also testified that the combination of incremental learning capability with reinforcement learning is a very effective approach to improving the performance of focused crawlers.

The rest of this paper is organized as follows. In section 2, we formalize focused crawling in a Q-learning framework. Section 3 describes the details of *IQ-Learning*. Experimental results and analysis are presented in Section 4. In Section 5 we draw some conclusions about our work and point out some directions for future work.

2 Focused Crawling as Q-learning

In [4], Rennie and et al. have presented a framework to formalize focused crawling as reinforcement learning. This section gives a brief summary of the main idea, and discusses the disadvantages of Rennies's method that can't perform incremental learning during the crawling process.

Reinforcement learning addresses the problem of how an autonomous agent can learn to choose optimal actions to achieve its goals through trial-and-error interactions with a dynamic environment [8]. Unlike supervised learning, its learner is never told the correct action for a specific state, but is simply told how good and bad the selected action is, expressed in the form of a scalar 'reward'. In the standard reinforcement learning model, the task of the agent can be defined as follows: the agent exists in an environment with a set of possible states S, and it can perform a set of actions A; when performing an action the state of the agent will change to another state defined by a transition function as: $\delta : S \times A \longrightarrow S$, and for each action the agent will get a reward defined by a reward function as: $r : S \times A \longrightarrow R$, the task of the agent is to learn a policy $\pi : S \longrightarrow A$ that maximizes the sum of its rewards over time (when the agent takes a set of actions).

Q-learning [8] is one of the most popular learning algorithms in reinforcement learning. It simplifies the policy learning as the estimation of the Q value for each state-action pair.The value $Q(s, a)$ (namely Q value) is defined to be the expected discounted sum of future rewards obtained by taking action a from state s and following an optimal policy thereafter. Once these values have been learned, the optimal action from any state is the one with the highest Q value. [8] gives a detailed explanation of Q-learning.

The goal of a focused crawler is to selectively seek out pages that are relevant to the predefined target topics. An ideal focused crawler retrieves the maximal set of relevant pages while simultaneously traversing the minimal number of irrelevant documents on the Web. From an agent perspective, a focused crawler is an intelligent agent that interacts with the Web environment to achieve its goal of getting maximum relevant pages. It's natural to fit the crawling process into a Q-learning framework: the crawling process is to search an optimal decision making strategy which can be modeled as a discounted function of state, action and reward, where the number of relevant pages collected is the state of the crawler, following a hyperlink is an action, and relevant page are immediate rewards (Q values).

To fit focused cawling in the Q-learning framework, two approximations are needed [4]: (1) the number of relevant pages on the Web is infinite and the state

is independent of it; (2) the distinction between the hyperlinks can be captured by the texts in the neighborhood of the hyperlinks, that is, the action is described as the link context of the corresponding hyperlink. With the first approximation, the state space becomes one single state, otherwise it will contain 2^n states, a very huge and insoluble number, where n is the actual number of relevant pages.

Based on the two approximations, the task of focused crawling in Q-learning framework is to follow a list of actions (hyperlinks) such that the cumulative Q value (the number of relevant pages) can be maximized. This can be further divided into three steps based on the two approximation as: (1) assigning the appropriate Q value to each hyperlink in the training set; (2) build Q value estimation model from the sample pair of hyperlink and its corresponding link context (such as the anchor text); (3) using the estimation model to compute a Q value for each candidate hyperlink (URL) in the URL queue to be crawled, where the Q value represents the crawling priority for the candidate URL.

In previous work, the step of assigning Q value is done by manually labeling which is too time-consuming and inefficient. Moreover, the estimation model remains the same and can't be self-adaptive in the crawling process. This is problematic for two reasons: (1) the initial samples for training the model are never sufficient to build an accurate model; (2) as the content of Web pages as well as the link structures for different topics are very diverse, and different target topics may require different crawling strategies, the static learned models cannot quickly adapt to the diverse Web environment. These problems are addressed in the *IQ-Learning* crawling algorithm, which effectively combines the incremental learning with *Q-learning*.

3 Focused Crawler Based on Incremental Q-Learning

3.1 An Incremental Q-Learning Framework

The *IQ-Learning* crawling algorithm is based on an incremental *Q-Learning* framework, as shown in Fig.1. In this framework, the crawling module gets URLs from the URL queue and downloads (crawls) the corresponding Web pages through HTTP/HTTPS protocols. The crawled pages will be analyzed to extract URLs which will be added to the URL queue as new candidate URLs to be crawled. Similar to general focused crawlers, the framework uses a page relevance estimator (page classifier) and a link predictor to make the crawler ***focus*** on the target topic (i.e., crawling selectively to get maximum relevant pages). The relevance of the crawled pages are computed by the page relevance estimator, which employs a TF*IDF [9] relevance evaluation model to compute a continuous relevance value for each crawled page. The Q value learner has two functions: one is to compute the Q value for each pages in the training set based on the relevance information provided by the page relevance estimator, the other is to build the Q value estimation model from the training samples through the mapping between link contexts and Q values. According to the Q value estimation model, the link predictor computes and assigns a Q value to each URL in the URL queue, where the Q values denote the visiting priority

(i.e. the probability of being relevant) for the candidate URLs, and the URLs with higher Q values will be crawled first.

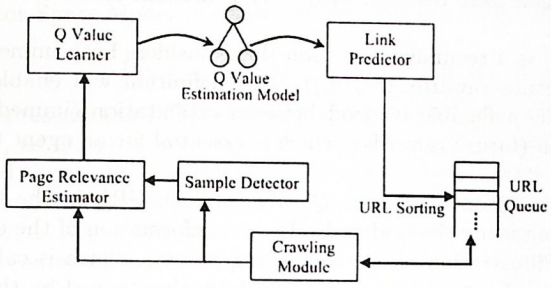


Fig. 1. An incremental *Q-Learning* framework for focused crawling.

There are two incremental learning processes in this framework. First, the page relevance estimator is an incremental learner. It is based on the 'soft' classification method from our work [10]. It needs only a few initial training samples and uses a sample detector to analyze the crawled pages to distill new samples automatically. The new samples are fed back to the page relevance estimator to learn a new relevance estimation model incrementally. This process will improve the accuracy of page relevance estimation over time, which has been testified in the previous work [10]. Details about this process were discussed in [10, 11].

Secondly, the Q value learner (and in turn the link predictor) also works in an incremental manner. Initially, the learner has a very limited training set. As the crawling proceeds, the crawled pages and their relevance information will be transferred to the Q value learner as the new learning resources. New training samples will be distilled based on predefined relevance thresholds. For 'old' training samples, their Q values will be updated as well. The Q value learner will learn from the training set, which is updated constantly during crawling, to improve its estimation model incrementally.

3.2 The Definition of An Updatable Q Value

The Q value for each new URL in the URL queue represents the crawling priority of the URL. Given a sample hyperlink (URL), the Q value learner computes the corresponding Q value according to the function defined as follows:

$$Q(u_i) = R(d_i) + \gamma Q(d_i), \qquad Q(d_i) = \frac{1}{n} \cdot \sum_{k=1, u_k \in d_i}^{n} Q(u_k) \tag{1}$$

where u_i is a crawled URL, d_i is the corresponding page for u_i. $Q(u_i)$ is the Q value for u_i, which is a dynamic value updated constantly during crawling.

$R(d_i)$ is the relevance of d_i with regard to the target topic, and is calculated by the page relevance estimator. u_k is a crawled child URL of d_i, that is, d_i links to the page corresponding to u_k. The value n is the total number of the children URLs of d_i which have been crawled. γ is a discount factor for converting the future rewards.

Formula (1) is a recursive definition that considers both immediate rewards ($R(d_i)$) and future rewards ($\gamma Q(d_i)$). This definition will enable the focused crawler to make a flexible tradeoff between exploitation (immediate rewards) and exploration (future rewards), which is essential for an agent to achieve the global optimal state.

According to formula (1), the Q value of each URL in the training set is calculated online using the updated relevance information of the crawled pages. Fig.2 gives an illustration of the computing process, which is called 'relevance feedback'. For each sample page, its Q value is determined by the Q values of its children pages recursively. When new pages are crawled (or the relevance values of the pages have been updated), the new relevance information will be propagated back to their ancestor pages along the link paths so that the ancestor pages can re-compute their Q values agilely. For instance, in Fig.2 the link path for page G (corresponding to the URL u) is $A \longrightarrow B \longrightarrow E \longrightarrow G$. The left part of Fig.2 denotes the link graph before the page G is crawled, while the right part shows the link graph after visiting G. During the feedback, the relevance of G is back-propagated along the link path $G \longrightarrow E \longrightarrow B \longrightarrow A$, so that the Q value for each page on the path is recalculated according to formula (1).

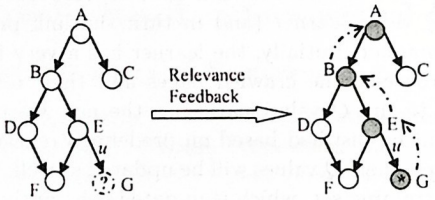


Fig. 2. An illustration of the relevance feedback along link path.

3.3 Mapping Link Context to Q Value

After computing the Q values for the sample URLs in the training set, the Q value estimator needs to build the Q value estimation model that can be used to predict the Q value for a candidate URL given the link context of the URL. The prediction of Q value is a regression problem because the Q value is continuous. Directly solving this regression problem is difficult as training samples are limitted. Similar to Rennies's method [4], we solve this problem by casting the regression problem into a classification prolbem through the discretization of

the Q value [8], and employ the Naive Bayes (NB) as the classification method to map link contexts into the discretized Q values.

For each training sample $\langle \Gamma(u_i), Q(u_i) \rangle$, the link context $\Gamma(u_i)$ for URL u_i is constructed by concatenating all the anchor texts from its parent pages. We use the Vector Space Model to represent $\Gamma(u_i)$ as a TF*IDF vector as:

$$\Gamma(u_i) = \langle \omega_{1i}, \omega_{2i}, \ldots, \omega_{ki}, \ldots, \omega_{ni} \rangle \tag{2}$$

where ω_{ki} represents the TF*IDF weight of the term t_k in the link context of u_i.

However, as $Q(u_i)$ is a continuous value, it can't be used directly by NB classifier and should be discretized and normalized as follows: all Q values are normalized into an interval [0,1] which is then discretized into k sub-intervals as: $C_1, C_2, \ldots C_i, \ldots, C_k$. Each sub-interval C_i represents a class, and the Q value of a class is computed as the average value of all the Q values in the sub-zone. After the discretization, the initial training sample $\langle \Gamma(u_i), Q(u_i) \rangle$ is represented as $\langle \Gamma(u_i), C_i \rangle$.

NB classifier is a probabilistic classifier based on the statistical independence assumption that features in learning instances are considered conditionally independent given the target label. For our mapping problem, the task of the classifier is to predict the class C^* for the link context of a candidate URL $\Gamma(u_i)$, where the Q value of the class C^* will be assigned to the URL u_i. In NB classifier, this can be formalized as: to find a class C^* for $\Gamma(u_i)$ such that the conditional probability $P(C^*|\Gamma(u_i))$ will be maximized:

$$C^* = \arg\max_{c_j} P(C_j|\Gamma(u_i)) = \arg\max_{c_j} P(C_j)P(\Gamma(u_i)|C_j) \tag{3}$$

Directly computing $P(\Gamma(u_i)|C_j)$ is difficult since the number of possible vectors for $\Gamma(u_i)$ is too high. However, according to the independence assumption of NB that the occurrence of term t_k in a link context is independent of the occurrence of any other term, the formula (3) can be replaced by formula (4) as:

$$C^* = \arg\max_{c_j} P(C_j)P(\Gamma(u_i)|C_j) = \arg\max_{c_j} P(C_j) \prod_{k=1}^{|\Gamma(u_i)|} P(\omega_{kj}|C_j) \tag{4}$$

Thereby, learning a NB classifier is to estimate $P(C_j)$ and $P(\omega_{kj}|C_j)$ from the training set. $P(C_j)$ can be calculated as the fraction of the Q value number of class C_j in the entire training set. The posterior probability $P(\omega_{kj}|C_j)$ is calculated as:

$$P(\omega_{kj}|C_j) = \frac{1 + \sum_{i=1}^{|c_j|} \omega_{ki}}{|V| + \sum_{i=1}^{|c_j|} \omega_i} \tag{5}$$

where $|V|$ is the vocabulary size and serves for Laplace smoothing of the non-occurring terms, ω_{ki} is the total TF*IDF weight of term t_k that occurs in the link context $\Gamma(u_i)$ of class C_j, and ω_i is the total weight of all term occurrences in the link context $\Gamma(u_i)$ of class C_j.

According to the above definitions, the NB learner will be trained incrementally during crawling, and the link predictor in Fig.1 uses the learned NB classification model to compute the Q value for each candidate URL as: the classfier classifies a candidate URL u_i into a certain class C^*, where the corresponding Q value of the class C^* is assigned to u_i as its Q value.

3.4 The Algorithm

The pseudo-code of *IQ-Learning* crawling algorithm is presented in Fig.3. The focused crawler starts from some seed URLs to crawl the Web selectively. At the initial stage, the page relevance estimator and Q value estimator need only a few training samples, which the user can easily provide. As the crawling process proceeds, new samples will be distilled automatically and fed back to the learners to build new estimation models, as shown in the step 11 and 13. As the page content and the link structure on the Web are diverse, this self-adaptive functionality will enable the focused crawler to learn an optimal crawling strategy quickly, so that it will not be restricted to the limited initial samples, and will greatly improve the performance and robustness of the system.

Ideally, the incremental learning should be performed continuously. However, as the learning process is computationally expensive, in practical implementations the crawler always executes the learning process in a batch mode, that is, every X pages (where X can be 100). This is a tradeoff between the crawling accuracy and the efficiency.

Input: K-keywords; W^*-initial samples; S-seed URLs;
maxDownload-maximum pages to be crawled
Output: D- the crawled page repository
1. Insert S into the URL queue H as the initial URLs for the crawling module;
2. Page relevance estimator learns from w^* initially;
3. while ((D.size$\leq$ ***maxDownload***) and (H.size$>$ 0)) do:
4. Fetch the URL u with highest Q value from H;
5. Crawl the page d corresponding to u;
6. $D = D \bigcup d$;
7. Extract new URLs U^* from d;
8. Insert U^* into H;
9. Page relevance estimator computes the relevance of d: $R(d)$;
10. Feed back $R(d)$ along the link paths to recalculate the Q values for the ancestor pages of d based on formula (1);
11. Re-training the naïve Bayes Q-value-mapping classifier with the new Q values got from step 10;
12. Using the new classifier to re-estimate Q values for candidate URLs in H;
13. Page relevance estimator learn incrementally from d using sample detector;
14. Sort the URL queue H based on the new Q values;
15. end while

Fig.3. The pseudo-code for the *IQ-Learning* algorithm.

4 Experiments

4.1 Experiment Setup

We have implemented the *IQ-Learning* algorithm in our focused crawler system *iSurfer*. A high-performance crawling module was used to download Web pages from the Web. The system also included an efficient HTML document parser to parse and analysis the pages. The well-known Porter stemming algorithm was used to stem English words. To experiment on Chinese Web pages, we used Maximized-Matched Chinese word segmentation algorithm to extract Chinese words from Chinese text blocks, as unlike English text there is no explicit delimiters like space between Chinese words.

Target topics were defined by the keywords and a few sample pages in our experiments. The initial samples were obtained through meta-searching the Yahoo! search engine. Keywords were sent to Yahoo! search, and the result pages were parsed to extract the hit URLs, which were presented to the system as the seed URLs as well as the initial samples for page relevance estimator.

In traditional information retrieval research, the precision and recall are the main evaluation metrics. However, it is difficult to get the recall for focused crawlers, as measuring the size of relevance set from the enormous Web is almost impossible. We employ the precision, which is named *'harvest rate'* in focused crawler community [1], as the main performance metric. Harvest rate is defined as the percentage of the crawled pages that are relevant w.r.t the target topics.

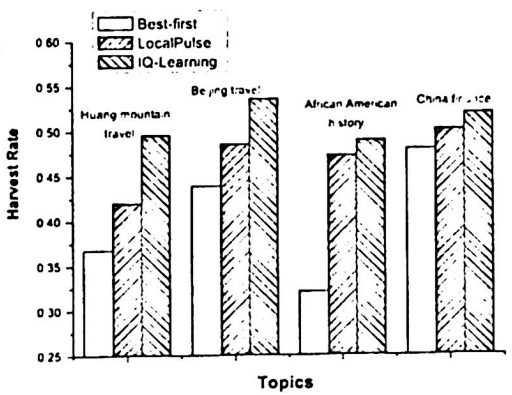
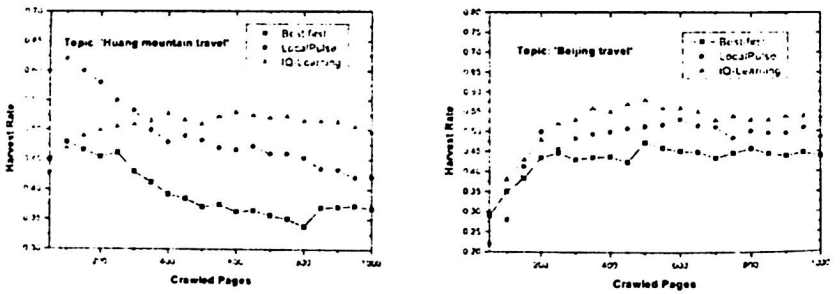


Fig. 4. The final harvest rates for the three algorithms.

4.2 Comparison Experiments

In our experiments, we compared *IQ-Learning* to the other two state-of-the-art focused crawling algorithms: (1) the best-first algorithm [12], which has been testified as one of the best-performed crawling strategies in previous comparison studies; (2) the *LocalPulse* algorithm [10, 11], which was used in *iSurfer* previously. *LocalPulse* has incremental learning capability, but it's not based on reinforcement learning, and it can't perform online relevance feedback along the link path, which affects its self-adaptive ability. The three algorithms were tested on more than 50 topics covering a variety of domains. These topics had diverse topical broadness, both in English and in Chinese. For each topic, crawlers started with 10 seed sample URLs.

Fig.4 shows the final harvest rates of the algorithms on some topics after crawling one thousand pages. Both *IQ-Learning* and *LocalPulse* outperformed the best-first crawler in all topics. This illustrates that incremental learning is very effective for improving the performance of focused crawler. Another important observation was that the *IQ-Learning* got higher harvest rate than *LocalPulse* although both of them can learn incrementally in the crawling process. To explain this phenomenon, we investigated their harvest rates during crawling.



(a) (b)

Fig. 5. The dynamic harvest rate movement of the three algorithms.

Fig.5 shows the dynamic harvest rates of the three algorithms on the Chinese topics 'Huang Mountain Travel' and 'Beijing Travel'. At the early stage of the crawling, *LocalPulse* got more relevant pages than *IQ-Learning*. However, as the crawling proceeds, *IQ-Learning* catch up with *LocalPulse* very quickly. This is due to the fact that *LocalPulse* employs a greedy strategy in selecting candidate URLs, which always focuses on the crawling area with more immediate rewards and does not consider future rewards, therefore it is very easily to stick into local optimal states. On the contrary, *IQ-Learning* makes a good tradeoff

between immediate rewards and future rewards, through which the crawler can get more relevant pages in the long run and can achieve global optimal states. This testifies that combining reinforcement learning with incremental learning is a more effective method for focused crawling.

4.3 The Tradeoff between Immediate Rewards and Future Rewards

According to formula (1), the parameter γ is a bias factor between the immediate rewards and future rewards. The value of it will greatly affect the performance of the system. We used different values of γ to do the crawling experiments. Fig.6 shows the experimental results on the topic 'Huang Mountain Travel'.

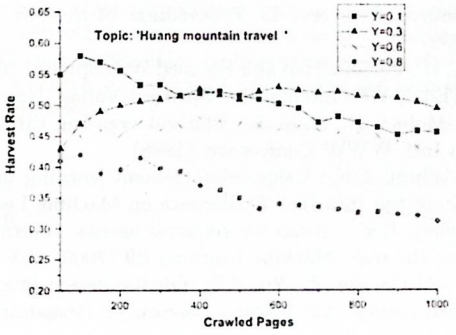


Fig. 6. The effect of the discount factor γ on *IQ-Learning*.

As the figure illustrates, the harvest rate will become higher in the early stage of the crawling process if γ is set to a smaller value, such as $\gamma = 0.1$ in Fig.6, which means the crawler is more 'greedy' and biased for the candidate URLs with immediate rewards. However, as the crawling process continues, the ones with bigger value of γ get higher performance because their early consideration of future rewards is paid for soon. On the other hand, the value of γ can't be set too high as well. Otherwise, the future rewards will be over-weighted w.r.t the immediate rewards, and the focused crawler may fall into topic-drift. As the figure shows, when γ is set to 0.8, the harvest rate of the crawler drops dramatically. In our experiences, a value between 0.3 0.4 for γ is a proper choice.

5 Conclusions

This paper presents *IQ-Learning*, a new focused crawling algorithm based on incremental *Q-learning*. Both the page relevance estimator and the Q value estimator are endowed with incremental learning ability to learn improved models

from the online crawled pages over time. With incremental learning, *IQ-Learning* can start with a few initial samples and self-adapt to diverse crawling environments, which will result in a more optimized crawling strategy. Our experimental results testifies that the combination of incremental learning with reinforcement learning is effective for improving the performance of focused crawler.

In the future, we plan to extend our work with a knowledge base which will enable the system to accumulate crawling knowledge from different crawling processes. We will also employ semantic-based Web page segmentation algorithms to enrich the link context with more relevant neighborhood text.

References

1. Chakrabarti, S., M., V.D.B., Dom, B.: Focused crawling: a new approach to topic-specific web resource discovery. In: Proceedings of the 8th International WWW Conference. (1999)
2. Chau, M., Chen, H.: Personalized and Focused Web Spiders. In Ning Zhong, Jiming Liu, Yiyu Yao (Eds), Web Intelligence. Springer-Verlag, New York. (2003)
3. Cho, J., Garcia-Molina, H., Page, L.: Efficient crawling through url ordering. In: Proc. of the 7th Intl. WWW Conference. (1998)
4. Rennie, J., McCallum, A.K.: Using reinforcement learning to spider the web efficiently. In: Proc. of the 16th Intl. Conference on Machine Learning. (1999)
5. Menczer, F., Belew, R.K.: Adaptive retrieval agents: Internalizing local context and scaling up to the web. Machine Learning **39** (2000) 203–242
6. Aggarwal, C.C., Al-Garawi, F., Yu, P.S.: On the design of a learning crawler for topical resource discovery. ACM Transactions on Information Systems, **19** (2001) 286–309
7. Chakrabarti, S., K., P., M., S.: Accelerated focused crawling through online relevance feedback. In: Proc. of the 11th Intl. WWW Conference. (2002)
8. Mitchell, T.: Machine Learning. McGraw-Hill, New York. (1997)
9. Baeza-Yates, R., Ribeiro-Neto: Modern Information Retrieval. ACM Press Series/Addison Wesley,, New York (1999)
10. Ye, Y., Ma, F., Lu, Y., Chiu, M., Huang, J.: Isurfer: A focused web crawler based on incremental learning from positive samples. In: Proceedings of the 6th Asia-Pacific Web Conference. (2004)
11. Ye, Y.: Topic-driven Web Resource Discovery: Models, Algorithms and Applications. PhD Thesis, Shanghai Jiao Tong University, Shanghai (2004)
12. Menczer, F., Pant, G., Srinivasan, P.: Topical web crawlers: Evaluating adaptive algorithms. ACM Transactions on Internet Technology **4** (2004) 378–419