Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence (AAAI-10)

Learning to Surface Deep Web Content

Zhaohui Wu, Lu Jiang, Qinghua Zheng, Jun Liu

MOE KLINNS Lab and SKLMS Lab, Xi'an Jiaotong University
No.28, Xianning West Road, Xi'an 710049, P.R.China
laowuz@gmail.com, roadjiang@yahoo.com, qhzheng@mail.xjtu.edu.cn, liukeen@mail.xjtu.edu.cn

Abstract

We propose a novel deep web crawling framework based on reinforcement learning. The crawler is regarded as an agent and deep web database as the environment. The agent perceives its current state and submits a selected action (query) to the environment according to Q-value. Based on the framework we develop an adaptive crawling method. Experimental results show that it outperforms the state of art methods in crawling capability and breaks through the assumption of full-text search implied by existing methods.

Introduction

Studies (Lawrence 1998) show deep web content is particularly important. Deep web surfacing or crawling enables leveraging existing search engine infrastructure hence adopted by most of crawlers, such as HiWE (Raghavan 2001), Hidden Web crawler (Ntoulas 2005), and Google's Deep Web crawler (Madhavan 2008). A critical challenge is how a crawler can automatically generate promising queries to carry out efficient crawling (Madhavan 2008), (Ntoulas 2005), (Barbosa 2005). Existing methods relied on an assumption that full-text search is provided by deep web databases, leading to that estimation techniques for full-text databases e.g. Zipf Law can hardly be applied to non full-text databases. We present a formal framework based on reinforcement learning for deep web crawling. A crawler is regarded as an agent and deep web database as the environment. The agent perceives its current state and submits an action (query) to the environment based on long-term reward. The environment responds by giving the agent some reward (new records) and changing it into the next state.

Framework and Algorithm

The process of deep web crawling is defined as a discrete Decision Process (S, A, P) consisting of a set of states S, a set of actions A and transition probabilities distribution P. A crawling process follows a specific issue policy $\pi : S \rightarrow A$. $s_t \in S$ represents the acquired portion of the deep web database records at the step t. $a(k) \in A$ (a for short) denotes a query to the database with keyword k, causing a transition from s_t to some successor state s_{t+1} with probability $p(s_{t+1}|a, s_t)$. D is the collection of all records residing in deep web database. $R(s_t, a) \subseteq D$ denotes the collection of records responded by execution of a at s_t. $\forall i, j$, $R(s_i, a) = R(s_j, a)$. The portion of new records retrieved by executing a at s_t is $R_{new}(s_t, a)$.

$$R_{new}(s_t, a) = R(s_t, a) \setminus \bigcup_{i=1}^{t-1} R(s_i, \pi(s_i)). \tag{1}$$

Reward function $r(s_t, a)$ is the reward received at the transition from state s_t to state s_{t+1} by executing action a.

$$r(s_t, a) = |R_{new}(s_t, a)| / |D|. \tag{2}$$

Actions cause a cost $cost(s_t, a) = t_a + t_r + |R_{new}(s_t, a)|$, where t_a is the cost of issuing an action and t_r is proportional to the average time of handling a response record. The expectation conditioned on the current state s and the policy π is called state-value function $V^\pi(s)$ of state s, computed from

$$V^\pi(s_t) = \sum_{i=0}^{h} \gamma^i r(s_{t+i}, \pi(s_{t+i})) \tag{3}$$

in which h is the step length and γ is the discount factor. π^* is an optimal policy defined as $V^{\pi^*}(s) \geq V^\pi(s)$ ($\forall s \in S$, $\forall \pi$). We write $V^{\pi^*} = V^*$. We can rewrite Q-function as:

$$Q(s_t, a) = r(s_t, a) + \max[\sum_{i=1}^{h} \gamma^i \times r(s_{t+i}, \pi(s_{t+i}))] \tag{4}$$

The formal definition of deep web crawling is defined as:
Problem 1 *Find such policy* $\pi^* \equiv \arg\max_\pi V^\pi(s_i)$ *under the constraint* $\sum_{i=0} cost(s_i, a) \leq cost_{MAX}$ *(*$\forall s_i \in S$*) that maximizes the accumulative reward value.*

Let $DF(s_t, a(k))$ ($DF(s_t, a)$ for short) denote number of documents containing keyword k in acquired record set $\bigcup_{i=1}^{t-1} R(s_i, \pi(s_i))$. Suppose at s_t, training set Tr is a set of executed actions, $|Tr| = t$. Similarly, candidate set C is a set of available action candidates for submission. Each action in either Tr or C is encoded in the same vector space. For an action a_i in C, its reward can be estimated as:

$$|\tilde{R}(s_t, a_i)| = \sum_{a_j \in Tr} \kappa(a_i, a_j) |R(s_t, a_j)| \tag{5}$$

where $\kappa(a_i, a_j)$ is a kernel function used to evaluate the distance between the given two actions. Three types of features (linguistic, statistical and HTML features) are incorporated to establish the feature space. We calculate each action's reward and Q-value as follows:

Theorem 1 *At state s_t, the reward of each action a in A can be calculated from*

$$r(s_t, a) = \frac{|R(s_t, a)| - DF(s_t, a)}{|D|} \tag{6}$$

Theorem 2 *At state s_t when $h=1$ the Q-value of an action a_i ($a_i, a_j \in C$, $a_i \neq a_j$) can be estimated as*:

$$Q(s_t, a_i) = r(s_t, a_i) + \max[DF(s_t, a_j)/|D| + r(s_t, a_j) - \bigcup_{i=1}^{t} R(s_i, \pi(s_i)) \times R(s_t, a_j)/|D|^2] \tag{7}$$

The adaptive RL crawling algorithm takes the current state and last executed action as input and outputs the next optimal action. It first calculates the reward of the last executed action and then updates the action set through Step 2 ~ 7, causing the agent to transit from current state s_t to successor state s_{t+1}. The training and candidate set are updated in accord with the new action set in Step 9. Step 10 ~ 13 estimates the reward and Q-value for each action in candidate set. The action that maximizes Q-value will be returned as the next optimal action.

Algorithm 1: Adaptive RL crawling algorithm

```
Input: $s_t$, $a_m$
Output: $\pi(s_{t+1})$
1:  calculate $a_m$'s reward following Eq. (6);
2:  for each document $a_i \in D$
3:     for each keyword $k$ in $a_i$ do
4:       if action $a(k) \notin A$ then $A = A \cup a(k)$;
5        else then update TF and DF of action $a(k)$;
6:     end for
7:  end for
8:  change the current state to $s_{t+1}$;
9:  $Tr = Tr \cup \{a_m\}$; $C = C \setminus \{a_m\}$;
10: for each $a_i \in C$
11:    update its reward using Eq. (5) and (6);
12:    calculate its Q-value using Eq. (7);
13: end for
14: return $\arg\max[Q(s_t, a)]$;
```

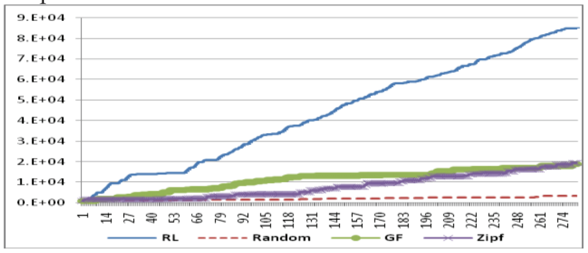
Experiments

To compare our RL method with existing methods, we choose the following three methods as the baseline:

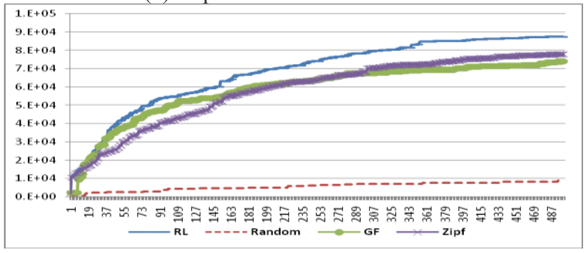
- Random: the reward of an action is assigned to a random float i.e. $Q(s_t, a) = random(1.0)$.
- Generic Frequency: the reward of an action is evaluated by the generic DF of the action at current state, i.e. $Q(s_t, a) = DF(s_t, a)$.
- Zipf: The size of response record set of each action is estimated by Zipf-Mandelbrot's Law (Mandelbrot 1982): $|R(s_t, a)| = \alpha(r + \beta)^{-\gamma}$,where α , β and γ are parameters and r is the DF rank of the action.

It is interesting to note that RL is more general compared with the baseline methods. If future reward of an action is ignored i.e. $h=0$ and the reward of an action is determined by a presumed distribution, the RL degenerates to Zipf, i.e. $Q(s_t, a_i) = r(s_t, a_i)$. Further if the acquired portion of an action is ignored too i.e. $R(s_t, a_i) = \Phi$, the RL degenerates to the GF, i.e. $Q(s_t, a_i) = DF(s_t, a_i)$.

To make results more intelligible, we roughly use harvest (number of actual retrieved records) and number of queries to evaluate the crawling. The experimental results of comparing our method with the baseline are displayed in Fig. 3, in which the y-axis denotes the number of acquired records and the x-axis represents the query number. In experiments step length was set to 1. The result shows that RL method is more efficient than the baseline methods on the experimental websites.



(a) Experiment on Baidu Baike



(b) Experiment on AbeBooks

Figure 3: Performance Comparisons with baseline methods

Acknowledgement

Research was supported by National High-Tech R&D Program of China under Grant No.2008AA01Z131, NSF of China under Grant Nos. 60825202, 60803079, 60633020, Key-Tech R&D Program of China under Grant Nos.2008BAH26B02, 2009BAH51B00, Cheung Kong Scholar's Program.

References

Barbosa, L., and Freire, J. Siphoning Hidden-Web Data through Keyword-Based Interfaces. 2004. In *SBBD*, 309-321.

Lawrence, S., and Giles, C. L. 1998. Searching the World Wide Web. *Science* 280:98–100.

Madhavan, J.; Ko, D.; Kot, L.; Ganapathy, V.; Rasmussen, A.; and Halevy, A. 2008. Google's Deep-Web Crawl. In *VLDB*, 1241-1252.

Mandelbrot, B. B. 1982. *The Fractal Geometry of Nature*. W. H. Freeman and Company.

Ntoulas, A.; Zerfos, P.; and Cho, J. Downloading Textual Hidden Web Content through Keyword Queries. 2005. In *JCDL*, 100-109.

Raghavan, S., and Garcia-Molina H. Crawling the Hidden Web. 2001. In *VLDB*, 129-138.

Wu, P.; Wen, JR.; Liu, H.; and Ma, WY. Query Selection Techniques for Efficient Crawling of Structured Web Source. 2006. In *ICDE*, 47-56.