Learnable Crawling: An Efficient Approach to Topic-specific Web Resource Discovery

Niran Angkawattanawit and Arnon Rungsawang
Massive Information & Knowledge Engineering
Department of Computer Engineering, Faculty of Engineering
Kasetsart University, Bangkok 10900 Thailand.
{g4365015, fenganr}@ku.ac.th

Abstract
The rapid growth of the Internet has put us into trouble when we need to find information in such a large network of databases. At present, using topic-specific web crawler becomes a way to seek the information. The main characteristic of topic-specific web crawler is trying to select and retrieve only the relevant web pages in each crawling processes. There are many previous researches focusing on the topic-specific web crawling. However, no one has ever mentioned about how the crawler does in the next crawling. In this paper, we present an algorithm that covers the detail of both the first and the next crawling. For efficient result of the next crawling, we keep the log of previous crawling to build some knowledge bases: seed URLs, topic keywords and URL prediction. These knowledge bases are used to build the experiences of the crawler to produce the result of the next crawling in a more efficient way.

1. Introduction

Topic-specific web crawler is a program used for searching information related the interested topics from World Wide Web. The main property of topic-specific crawling is that the crawler do not need to collect all web pages, but selects and retrieves relevant pages only. Because the crawler is only computer program, it does not make some relevant web page decisions as good as human do. Therefore, most previous works focus on how to create an algorithm for selecting relevant web page. In the literature, there are many researches mentioning about the selection algorithms of topic-specific crawling. For example, they are the breadth-first search [12], pagerank [4,8,15], focused crawling [17,10], shark-search algorithm [11], adaptive agent [2,3,9], reinforcement learning [6], artificial life agent[13], arbitrary predicate [1], WTMS [18,19], etc. The main purpose of these algorithms is to gather the most relevant web pages as possible. Therefore, the assessment of topic-specific web crawling is showed by the relevant graph which represents the proportion of the number of relevant web pages and the total number of downloaded web pages. If this proportion is high in early period of the crawling process, this means that the crawler can collect more relevant web pages than irrelevant ones.

Although, the above-mentioned algorithms are very efficient, they only provide the detail of the first crawling. However, to freshen the information, the crawling processes need to be done over and over. The problem we concentrate here is whether the next crawling process can be done in an incremental way, or how we can keep track the changing of web pages. This paper presents an algorithm that covers the detail of the first and the consecutive crawling. We add a learning ability from previous crawling to improve the efficiency of consecutive crawling processes. We argue that the crawler should have three kinds of learnable knowledge as follows: seed URLs, topic keywords and URL prediction. For seed URLs, the crawler needs some good URLs which point to many relevant web pages. For topic keywords, the crawler needs some appropriate keywords of interested topics. For URL prediction, the crawler needs to predict the undownloaded pages. These knowledge bases are considered as the experiences of the crawler. Therefore, the crawler which includes these knowledge bases should provide an efficient result during the next crawling.

This paper is structured as follows. Section 3 shows a characteristic of our learnable crawler and an algorithm exposing the first crawling and the crawling processes. In Section 4, we analyze the learning ability of the crawler. Section 5, we show how the consecutive crawls can be done more efficiently. Finally, Section 6 concludes the paper.

2. Learnable Web Crawler

In this section, we will shortly explain a characteristic of the web crawler, learning ability. We build some knowledge bases from the previous crawling. These knowledge bases are: seed URLs, topic keywords and URL prediction. Although, the previous crawling may not be efficient, the next crawling may be more efficient when the crawler uses the knowledge bases as its experience. Seed URLs support the crawler to collect as many relevant web pages as possible. Topic keywords support the crawler to recognize appropriate keywords matching the topic. URL prediction supports the crawler to predict the relevancy of the pages of unvisited URLs.

2.1 Seed URLs

Seed URLs are very important for the crawler. If these URLs do not contain relevant web pages, the crawler has less probability to find many other relevant ones. While the crawler does not have any knowledge base, the seed URLs of the interested topic can be deduced from a search result of a selected search engine. Normally, search engines will rank their search results by some popular ranking techniques such as PageRank [4,8,15], HITS [5,16], or BHITS [7], using authority and hub scores. If a page has the highest authority score, it will be ranked to the first result. Therefore, this set of seed URLs deduced from the selected search engines are the pages of high authority scores and are probably to be
extracted from <Title></Title> tag is frequently occurred in
the most popular keywords in the topic. The keywords
keyword sets. The crawler will memorize these keywords as
each relevant page. We will give the crawler some frequent
within the <Title></Title> tag and the <A></A> tag from
a set of relevant web pages. We will extract some texts
to build the knowledge base of topic keywords, we need
To solve the above problem, we really want a set of seed
URLs in which each URL has both a high hub score, and a
high authority score. A good hub page will point to many
relevant pages. Therefore, to build the knowledge base of
the seed URLs, we need a number of web pages from the
previous crawling to compute hub and authority scores of
pages using BHITS algorithm. We do not show the BHITS
method here because the space limitation.

2.2 Topic Keywords

Normally, a document having its content related to a
topic is composed of a set of keywords which frequently
appear in the topic. For example, a tcp/ip document is often
contained by the following keywords: “tcp, ip, header,
packet” or “protocol”, etc. The crawler uses these keywords
as a clue or guideline in topic-specific crawling. Sometimes,
the crawler misses some relevant pages because the crawler
has too little significant keywords or has not enough popular
keywords of the topic to be considered. For example, the
crawler does the topic-specific crawling of tcp/ip documents
using only the following keywords: “tcp” and “ip”, this
means that any page containing the words “tcp” or “ip” will
be retrieved by the crawler. On the other hand, any page
which does not contain the words “tcp” or “ip” will not be
retrieved although it contains the words “header, packet” or
“protocol”. Therefore, the topic keywords is an important
factor and should be built as a knowledge base for the
crawler.

To build the knowledge base of topic keywords, we need
a set of relevant web pages. We will extract some texts
within the <Title></Title> tag and the <A></A> tag from
each relevant page. We will give the crawler some frequent
keyword sets. The crawler will memorize these keywords as
the most popular keywords in the topic. The keywords
extracted from <Title></Title> tag is frequently occurred in
the relevant pages while the keywords extracted from
<A></A> tag is frequently occurred in the pages which
point to relevant pages.

2.3 URL Prediction

In the first crawling, content of a page is known when the
crawler has finished downloading that page. While the page
is not downloaded, the crawler is guided by the anchor text
to decide whether the page should be retrieved. Sometimes
the anchor text does not relate to the topic we are interested
in, but the content of the page is relevant. On the contrary,
the anchor text does relate to the topic, but the content of the
page is not relevant. In other words, the crawler is blind
during the first crawling. When comparing with human
learning ability, before human have some skills and
knowledge, they have never had any of them. Human will
learn some experiences to make skills and knowledge. Thus,
we need our crawler to have an ability to learn something as
human. Although, the crawler is blind in first crawling, the
crawler can gain some experiences by memorizing what it
has seen. The benefit of content recognition will then be
utilized in the next crawling. When the crawler must decide
whether a URL contains a relevant page, it will get the
previous seen content of the page from its database to predict
the new content. We refer to this technique as the
URL prediction. URL prediction helps the crawler to
strongly decide whether a URL should be downloaded. To
predict the content of the URL, we compute the page
similarity [14] between the content of the previous crawled
pages and the interested topic.

2.4 The Algorithm

The algorithm of our learnable web crawler is inspired
from the shark-search algorithm described in [11] and from
our previous research [13]. We may separate our crawling
algorithm into two parts: crawling with no knowledge bases
and crawling with knowledge bases. Crawling with no
knowledge bases is used in the first crawling since the
crawler has not had any knowledge yet. Crawling with
knowledge base is used in the consecutive crawling. These
knowledge bases are continuously built from the previous
crawling.

1 Crawling_with_no_KB (topic) {
2 starting_urls := Search (topic, t);
3 keywords := topic;
4 foreach url (starting_urls) {
5 url_topic := url.title + url.description;
6 url_score := sim (keywords, url_topic);
7 enqueue (url_queue, url, url_score);
8 }
9 while (#url (url_queue) > 0) {
10 url := dequeue_url_with_max_score (url_queue);
11 page := fetch_new_document (url);
12 page_score := sim (keywords, page);
13 foreach link (extract_urls (page)) {
14 link_score := a·sim(keywords, link.anchortext)
+ (1-a)·page_score;
15 enqueue (url_queue, link, Link_score);
16 }
17 }
18 }

Figure 1. Pseudo-code of Crawling with no Knowledge bases algorithm.

1 Crawling_with_KB (KB, topic) {
2 starting_urls := get_start_url(KB,topic,t);
3 keywords := get_topic_keyword(KB,topic);
4 foreach url (starting_urls) {
5 url_score := get_pred_score(KB,topic,url);
6 enqueue (url_queue, url, url_score);
7 }
8 while (#url (url_queue) > 0) {
9 url := dequeue_url_with_max_score (url_queue);
10 page := fetch_new_document (url);
11 page_score := sim (keywords, page);
12 foreach link (extract_urls (page)) {
13 pred_link_score := get_pred_score(KB,topic,url);
14 link_score := a·pred(link_score)
+ (1-a)·page_score;
15 enqueue (url_queue, link, Link_score);
16 }
17 }
18 }

Figure 2. Pseudo-code of Crawling with Knowledge bases algorithm.

In Figure 1, to begin the crawling with no knowledge
bases, we assign a topic to the crawler. The crawler gets t
starting URLs from the search result of a selected search
engine. The crawler will then rank the starting URLs using
the similarity scores of each title and description text. Since
there is no knowledge base, a topic keyword will then be
used as the name of the topic. When the crawler gets a set of
new URLs, it has to predict whether those URLs contain
relevant pages or not. This can be done by computing two scores: the similarity score of the anchor text and the similarity score of the parent page. Considerably, if the crawler does not have any knowledge, its crawling process may not be efficient. Therefore, we suppose that the next crawling should be better since the crawler will gain more experience from the first crawling. We denote $KB$ to be the knowledge bases of the crawler.

In Figure 2, to begin the crawling with knowledge bases, we assign a topic to the crawler. The crawler gets starting URLs from the knowledge bases using the get_start_URL function. These starting URLs are good hubs which point to many other relevant pages. Next, the crawler gets the keywords matching to the topic. Topic keywords are obtained from the get_topic_keyword function. When the crawler gets a set of new URLs, it has to predict whether those URLs contain relevant pages or not by computing three scores: the similarity score of the anchor text, the similarity score of the parent page, and the predicting score of the URLs. The crawler computes the predicting score using the get_pred_score function. For the consecutive crawling, the crawler should use this crawling with knowledge bases algorithm as it will be more efficient.

3. Learning Analysis

A good crawler should collect many relevant pages at the beginning of the crawling process. We can evaluate the crawling efficiency using a relevant graph shown in Figure 3 below. The relevant graph records the relationship between the number of collected pages and the number of relevant pages of each crawling. The x-axis is the percentage of the collected pages, and the y-axis is the percentage of the relevant pages.

We call three curves in Figure 3, the learning curves. The curve No.1 shows a bad crawling example since at the beginning of crawling process the crawler can retrieve less relevant pages comparing to the total pages being collected. The curve No.2 is an assuming crawling example that we will represent it as a base line to separate between the good crawling example and the bad one. Finally, the curve No.3 is a good crawling example since at the beginning of crawling process the crawler can retrieve most of the relevant pages comparing to the total pages being collected.

We can see the learning tendency of a crawler during consecutive crawling by examining its learning curves. We define two types of learning: 1) Negative Learning and 2) Positive Learning.

A crawler has negative learning ability when it can collect less relevant pages at the beginning of the crawling process during the consecutive crawling. For example, in Figure 4, the percentage of relevant pages the crawler can retrieve decreases from the first, the second and the third crawling, respectively. Evidently, this is not what we want the crawler to do. However, this can be happened when the content of previous collected pages is changed from relevant to irrelevant one, and the fault URL prediction results the crawler in missing many relevant pages.

On the other way, a crawler has positive learning ability when it can collect much more relevant pages at the beginning of the crawling process during the consecutive crawling. For example, in Figure 5, the percentage of relevant pages the crawler can retrieve increases from the first, the second and the third crawling, respectively. This result can be come from using the good starting URLs, good topic keywords, and the fact that the URL prediction works as it should be. Good starting URLs should be the best hubs which point to many other relevant pages, so that the crawler has a high probability to collect many relevant pages at the beginning of the crawling. Good topic keywords are those words that are more specific with a certain topic. This makes the crawler knows which pages are relevant or not. Finally, good URL prediction can results the crawler in collecting the pages having irrelevant anchor text but relevant content.

4. Efficient Learning

To reduce the network bandwidth usage, an efficient crawling of a topic-specific web crawler is preferred. A crawler has efficient learning ability if it can collect many relevant pages at the beginning of the crawling processes.
during the consecutive crawling. Figure 6 shows an example of an efficient learning crawler. The tendency of the learning ability, illustrated by its learning curves, increases from the first till the fifth and the sixth crawling, respectively. The crawler can retrieve more than 90% of relevant pages when it visits less than 25% of the total pages collected from the first crawling.

![Figure 6. Efficient Crawling.](image)

**5. Conclusion**

This paper presents a learnable topic-specific web crawler. There are three main parts discussed in the paper: crawler’s knowledge bases, crawling algorithm, and analysis of crawler learning ability. The knowledge bases of the crawler are incrementally built from the log of previous crawling. For efficient result of the next crawling, we present three knowledge bases: starting URLs, topic keywords, and URL prediction. Good starting URLs support the crawler to collect as many relevant web pages as possible. Good topic keywords support the crawler to recognize appropriate keywords matching the required topic. Good URL prediction supports the crawler to predict the relevancy of the content of unvisited URLs. Crawling algorithm has been separated into two parts: crawling with no knowledge bases and crawling with knowledge bases. Crawling with no knowledge bases is used in the first crawling for Internet exploration. The information gathered from the first crawling has been accumulated to be the experience of the crawler, i.e. the knowledge bases, during the consecutive crawling. Crawling with knowledge bases should be used in the next crawling for more efficient result and better network bandwidth utilization.

**6. Reference**