IJREAT International Journal of Research in Engineering & Advanced Technology, Volume 2, Issue 2, Apr-May, 2014
ISSN: 2320 – 8791 (Impact Factor: 1.479)
www.ijreat.org

URL Mining Using Web Crawler in Online Based Content Retrieval

M.VijayaLakshmi[1], Mr.P.Senthil Kumar[2]

[1]II M.E Student, Department of Computer Science and Engineering, S.Veerasamy Chettiar College of Engineering and Technology, Puliangudi-627 855

[2]Assistant Professor, Department of Computer Science and Engineering, S.Veerasamy Chettiar College of Engineering and Technology, Puliangudi-627 855

Abstract

A supervised web scale forum crawler is a crawling process of forum crawler under supervision(Focus). The main aim of Focus is to crawl related content from the web with minimal overhead and also detect the duplicate links.Forums can contain different layouts or styles and are powered by a variety of forum software packages. Focus take six path from entry page to thread page. It helps the frequent thread updating in forum. It's main purpose is reduce the web forum crawling problem to a URL-type recognition problem.The Focus consists of two parts learning part and online crawling part.The learning part is automatically constructed URL training sets and then online crawling part to crawl all threads efficiently. The accurate and effective regular expression patterns of implicit navigation paths from automatically created training sets using aggregated results from weak page type classifiers.An effective forum entry URL discovery method to ensure the high coverage. The forum crawler should start crawling forum pages from forum entry URLs to thread URLs. The implicit EIT-like path also apply to other User Generated Content (UGC).

Keywords: *Component, formatting, style, styling, insert.*

1. Introduction

Data mining (the analysis step of the "Knowledge Discovery in Databases" process, or KDD),an interdisciplinary subfield of computer science, is the computational process of discovering patterns in large data sets involving methods at the intersection of artificial intelligence, machine learning, statistics, and systems. The overall goal of the data mining process is to extract information from a data set and transform it into an understandable structure for further use. Aside from the raw analysis step, it involves database and data management aspects data model and inference considerations ,interestingness metrics, post-processing of discovered structures, visualization, and online updating.The term is a buzzword, and is frequently misused to mean any form of large-scale data or information processing (collection, extraction, warehousing, analysis, and statistics) but is also generalized to any kind of computer decision support system, including artificial intelligence, machine learning, and business intelligence. The actual data mining task is the automatic or semi-automatic analysis of large quantities of data to extract previously unknown interesting patterns such as groups of data records (cluster analysis), unusual records (anomaly detection) and dependencies (association rule mining). This usually involves using database techniques such as spatial indices. These patterns can then be seen as a kind of summary of the input data, and may be used in further analysis or, for example, in machine learning and predictive analytics. For example, the data mining step might identify multiple groups in the data, which can then be used to obtain more accurate prediction results by a decision support system. Neither the data collection and data preparation, nor result interpretation and reporting are part of the data mining step, but do belong to the overall KDD process as additional steps.

The Knowledge Discovery in Databases (KDD) is commonly defined with the following stages. (i)Selection, (ii) Pre-processing, (iii) Transformation, (iv) Data Mining, (v) Interpretation/Evaluation. It exists, however, in many variations on this theme, such as the Cross Industry Standard Process for Data Mining (CRISP-DM) which defines five phases: Business Understanding, Data Understanding, Data Preparation, Modeling, and Evaluation.

Before data mining algorithms can be used, a target data set must be assembled. As data mining can only uncover patterns actually present in the data, the target data set must be large enough to contain these patterns while remaining concise enough to be mined within an acceptable time limit. A common source for data is a data mart or data warehouse. Pre-processing is essential to analyze the multivariate data sets before data mining. The target set is then cleaned. Data cleaning removes the observations containing noise and those with missing data.

Anomaly detection (Outlier/change/deviation detection) – The identification of unusual data records, that might be interesting or data errors that require further investigation.

Association rule learning (Dependency modeling) Searches for relationships between variables. For example a supermarket might gather data on customer purchasing habits. Using association rule learning, the supermarket can determine which products are frequently bought together and use this information for marketing purposes. This is sometimes referred to as market basket analysis.

Clustering – is the task of discovering groups and structures in the data that are in some way or another "similar", without using known structures in the data.

Classification – is the task of generalizing known structure to apply to new data. For example, an e-mail program might attempt to classify an e-mail as "legitimate" or as "spam".

Regression – Attempts to find a function which models the data with the least error.

Summarization – providing a more compact representation of the data set, including visualization and report generation.

There have been some efforts to define standards for the data mining process, for example the 1999 European Cross Industry Standard Process for Data Mining (CRISP-DM 1.0) and the 2004 Java Data Mining standard (JDM 1.0). Development on successors to these processes (CRISP-DM 2.0 and JDM 2.0) was active in 2006, but has stalled since. JDM 2.0 was withdrawn without reaching a final draft

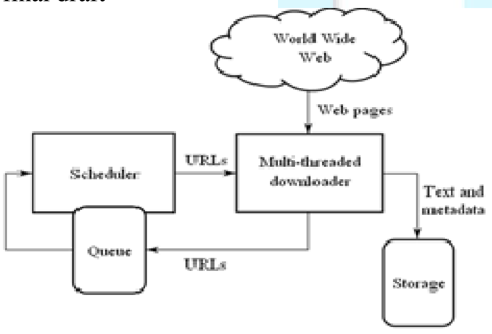

Fig 1 Architecture of focused Crawling

for use in predictive analytics – the key standard is the Predictive Model Markup Language (PMML), which is an XML-based language developed by the Data Mining Group (DMG) and supported as exchange format by many data mining applications. As the name suggests, it only covers prediction models, a particular data mining task of high importance to business applications. However, extensions to cover (for example) subspace clustering have been proposed independently of the DMG. A Web crawler is an Internet boot that systematically browses the World Wide Web, typically for the purpose of Web indexing. A Web crawler may also be called a Web spider, an ant, an automatic indexer, or (in the FOAF software context) a Web scutter.

Web search engines and some other sites use Web crawling or spidering software to update their web content or indexes of others sites' web content. Web crawlers can copy all the pages they visit for later processing by a search engine that indexes the downloaded pages so that users can search them much more quickly. Crawlers can validate hyperlinks and HTML code. They can also be used for web scraping see also data-driven programming.

A Web crawler starts with a list of URLs to visit, called the seeds. As the crawler visits these URLs, it identifies all the hyperlinks in the page and adds them to the list of URLs to visit, called the crawl frontier. URLs from the frontier are recursively visited according to a set of policies. The large volume implies that the crawler can only download a limited number of the Web pages within a given time, so it needs to prioritize its downloads. The high rate of change implies that the pages might have already been updated or even deleted.

The number of possible crawlable URLs being generated by server-side software has also made it difficult for web crawlers to avoid retrieving duplicate content. Endless combinations of HTTP GET (URL-based) parameters exist, of which only a small selection will actually return unique content. For example, a simple online photo gallery may offer three options to users, as specified through HTTP GET parameters in the URL. If there exist four ways to sort images, three choices of thumbnail size, two file formats, and an option to disable user-provided content, then the same set of content can be accessed with 48 different URLs, all of which may be linked on the site. This mathematical combination creates a problem for crawlers, as they must sort through endless combinations of relatively minor scripted changes in order to retrieve unique content.

URL normalization Crawlers usually performs some type of URL normalization in order to avoid crawling the same resource more than once. The term URL normalization, also called URL canonicalization, refers to the process of modifying and standardizing a URL in a consistent manner. There are several types of normalization that may be performed including conversion of URLs to lowercase, removal of "." and ".” segments, and adding trailing slashes to the non-empty path component.

Some crawlers intend to download as many resources as possible from a particular web site. So path-ascending crawler was introduced that would ascend to every path in each URL that it intends to crawl. For example, when given a seed URL of http://llama.org/hamster/monkey/page.html, it will attempt to crawl /hamster/monkey/, /hamster/, and /. Cothey found that a path-ascending crawler was very effective in finding isolated resources, or resources for which no inbound link would have been found in regular crawling.

1.1 Focused Crawling

The importance of a page for a crawler can also be expressed as a function of the similarity of a page to a given query. Web crawlers that attempt to download pages that are similar to each other are called focused crawler or topical crawlers. The concepts of topical and focused crawling were first introduced by Menczer and by Chakrabarti et al. The main problem in focused crawling is that in the context of a Web crawler, we would like to be able to predict the similarity of the text of a given page to the query before actually downloading the page. The architecture of focused crawling is shown in fig 1. A possible predictor is the anchor text of links; this was the approach taken by Pinkerton in the first web crawler of the early days of the Web. Diligenti et al. propose using the complete content of the pages already visited to infer the similarity between the driving query and the pages that have not been visited yet. The performance of a focused crawling depends mostly on the richness of links in the specific topic being searched, and a focused crawling usually relies on a general Web search engine for providing starting points.

A crawler must not only have a good crawling strategy, as noted in the previous sections, but it should also have a highly optimized architecture. While it is fairly easy to build a slow crawler that downloads a few pages per second for a short period of time, building a high-performance system that can download hundreds of millions of pages over several weeks presents a number of challenges in system design, I/O and network efficiency, and robustness and manageability. Web crawlers are a central part of search engines, and details on their algorithms and architecture are kept as business secrets. When crawler designs are published, there is often an important lack of detail that prevents others from reproducing the work. There are also emerging concerns about "search engine spamming", which prevent major search engines from publishing their ranking algorithms.
Duplicate documents in the World Wide Web adversely affect crawling, indexing and relevance, which are the core building blocks of web search. To present a set of techniques to mine rules from URLs and utilize these learnt rules for de-duplication using just URL strings without fetching the content explicitly. The crawl logs utilizing clusters of similar pages are extracted from specific rules from URLs belonging to each cluster. Preserving each mined rules for de-duplication is not efficient due to the large number of specific rules. A machine learning technique to generalize the set of rules, which reduces the resource foot-print to be usable at web scale is used. The rule extraction techniques are robust against web-site specific URL conventions.

To send in hypertext, Google is designed to crawl and index the Web efficiently and produce much more satisfying search results than existing systems. The prototype with a full text and hyperlink database of several pages is available. A search engine is a challenging task. Search engines index tens to hundreds of millions of Web pages involving a comparable number of distinct terms. They answer tens of millions of queries every day. Despite the importance of large-scale search engines on the Web, very little academic research has been done on them. Furthermore, due to rapid advance in technology and Web proliferation, creating a Web search engine today is very different from three years ago.It provides an in-depth description of our large-scale Web search engine - the first such detailed public description we know of to date. Apart from the problems involved with using the additional information present in hypertext to produce better search results. To address how to build a practical large-scale system which can exploit the additional information is present in hypertext.

2. Proposed System

Generic crawlers process each page individually and ignore the relationships between such pages. These relationships should be preserved while crawling to facilitate downstream tasks such as page wrapping and content indexing. INTERNET forums (also called web forums) are important services where users can request and exchange information with others. Generic crawlers, which adopt a breadth-first traversal strategy, are usually ineffective and inefficient for forum crawling. This is mainly due to two non crawler friendly characteristics of forums duplicate links and uninformative pages and page-flipping links. A forum typically has many duplicate links that point to a common page but with different URLs. A generic crawler that blindly follows these links will crawl many duplicate pages, making it inefficient. A forum also has many uninformative pages such as login control to protect user privacy or forum software specific FAQs.

Web crawlers typically identify themselves to a Web server by using the User-agent field of an HTTP request. Web site administrators typically examine their Web servers' log and use the user agent field to determine which crawlers have visited the web server and how often. The user agent field may include a URL where the Web site administrator may find out more information about the crawler. Examining Web server log is tedious task therefore some administrators use tools such as Crawl Track or SEO Crawlytics to identify, track and verify Web crawlers. Spam bots and other malicious Web crawlers are unlikely to place identifying information in the user agent field, or they may mask their identity as a browser or other well-known crawler.

It is important for Web crawlers to identify themselves so that Web site administrators can contact the owner if needed. In some cases, crawlers may be accidentally trapped in a crawler trap or they may be overloading a Web server with requests, and the owner needs to stop the crawler.

Identification is also useful for administrators that are interested in knowing when they may expect their Web pages to be indexed by a particular search engine.

2.1 Crawling the Deep Web

A vast amount of web pages lie in the deep or invisible web. These pages are typically only accessible by submitting queries to a database, and regular crawlers are unable to find these pages if there are no links that point to them. Google's Sitemaps protocol are intended to allow discovery of these deep-Web resources.

Deep web crawling also multiplies the number of web links to be crawled. Some crawlers only take some of the <a href="URL">-shaped URLs. In some cases, such as the Google bot, Web crawling is done on all text contained inside the hypertext content, tags, or text.

Strategic approaches may be taken to target deep-Web content. With a technique called screen scraping, specialized software may be customized to automatically and repeatedly query a given Web form with the intention of aggregating the resulting data. Such software can be used to span multiple Web forms across multiple Websites. Data extracted from the results of one Web form submission can be taken and applied as input to another Web form thus establishing continuity across the Deep Web in a way not possible with traditional web crawlers.

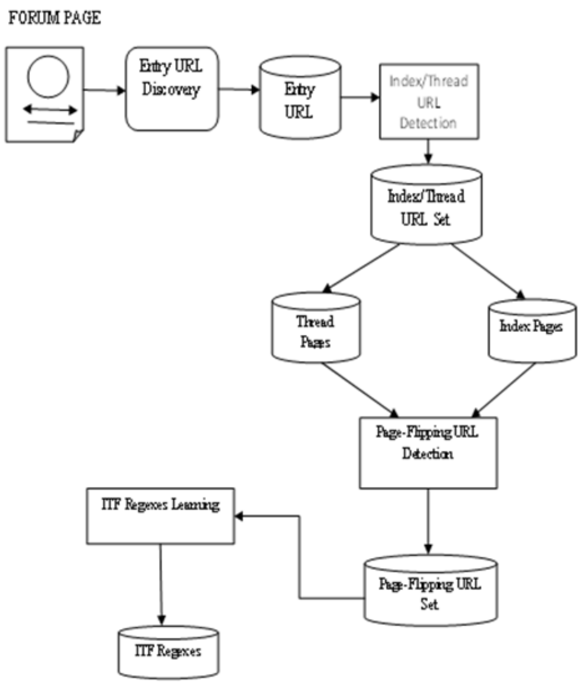

Fig 2 Architecture of the Proposed System

Forum Crawler Under Supervision (Focus), a supervised web scale forum crawler, to address these challenges. The goal of Focus is to crawl relevant content that is user posts, from forums with minimal overhead. Forums exist in many different layouts or styles and are powered by a variety of forum software packages, but they always have implicit navigation paths to lead users from entry pages to thread pages. The challenge of forum crawling is then reduced to a URL type recognition problem. To learn URL patterns, Index-Thread-page-Flipping (ITF) regexes recognizing these three types of URLs from as few as five annotated forum packages and apply them to a large set of 160 unseen forums packages. Note that we specifically refer to "forum package" rather than "forum site. The proposed system can be divided into the following modules.

i) Internal URL

ii) In order Crawled
iii) In order of Size
iv) External URL
v) Bad URL

2.1.1In Order Crawled

The pages were found within the site. The size is calculated by getting value of the Length of the text of the response text. This is the order in which they were crawled. This module contains the page size, view state size and list of internal URLs.

2.1.2 In Order of Size

The pages are found within the site. The size is calculated by getting value of the Length of the text of the response text. This is the order in terms of total page size. This module also contains the page size, view state size and list of internal URLs.

2.1.3 External URL

The pages are link to the outside of the site, a hyperlink on a Web page that points to the web page on a different Web site. On a blog, a link is typically considered external if it points to another blog, even though both blogs are hosted on the same blog site. It will search the link in different pages in a different web site.

Bad URL

The crawler to find the URL in the overall site and to display all corresponding equaling lists in a web site. Non proper links will be displayed in this Bad URL list. This contains the irresponsive links and proper connectionless links. This links will not be used in our web sites.

Algorithm

Index Url Thread Url Detection Algorithm

```
Input: sp:an entry page or index page
Output: it_group:a group of index/thread URLs
1:  let it_group be ;data
2:  url_groups=Collect URL groups by aligning HTML DOM tree of sp;
3: foreach ug in url_groups do
4: ug.anchor_len=Total anchor text length in ug;
5: end foreach
6: it_group=arg max(ug.anchor_len) in url_groups;
7: it_group.DstPageType=Majority page type of the destination pages of  URLs  in ug;

8:  if it_group.DstPageType is INDEX_PAGE
9:  it_group.UrlType=INDEX_URL;
10: else if it_group.DstPageType is THREAD_PAGE
11: it_group:UrlType=THREAD_URL;
12: else
13: it_group
14: end if
15: return it_group;
```

Page-Flipping URL Detection Algorithm

```
Input: sp:an index page or thread page
Output: pf_group:a group of page-flipping URLs
1: let pf_group be ϕ
2: url_groups=Collect URL groups aligning HTML DOM tree of sp;
3: foreach ug in url_groups do
4: if the anchor texts of ug are digit strings
5: pages=Download(URLs in ug);
6: if pages have the similar layout to sp and ug appears at same location of pages as in sp
7: pf_group=ug;
8: break;
9: end if
10: end if
11:end foreach
12:if pf_group is ϕ
13: foreach url in outgoing URLs in sp
14: p=Download(url);
15: pf_url=Extract URL in p at the same location as url in sp;
16: if pf_url exists and pf_url.anchor= =url.anchor and pf_url!=url.UrlString
17: Add url and cand_url into pf_group;
18: break;
19: end if
20: end foreach
21: end if
22: pf_group.UrlType=PAGE_FLIPPING_URL;
23: return pf_group;
```

Entry URLDiscovery Algorithm

```
Input:  url:a URL pointing to a page from a forum
Output:  entry_url:Entry URL of this forum
1: b_url=GetNaiveEntryUrl(url);
2: p=Download(url);
3: urls=Extract outgoing URLs in p that start with b_url;
4: samp_urls=Randomly sample a few URLs from urls;
5: Add the host of url into samp_urls;
6: foreach u in samp_urls do
7: p=Download(u);
```

```
8: urls= urls  {outgoing URLs in p starting with b_url};
9: end foreach
10: let entry_url be b_url,index_urls be ϕ count be 0;
11: foreach u in urls do
12: if u is in index_urls continue;ϕ
13: p=Download(u);
14: i_urls=Detect index URLs in p;
15: index_urls=index_urls  i_urls;
16: if count<li_urlsl
17: count=li_urlsl;
18: entry_url=u;
19: end if
20: end foreach
21: return entry_url;
```

3. Experimental Results

To determine the effectiveness of the proposed system, we selected 100 different forum software packages from different forums. For each package, there was a forum powered by it. Therefore, there were 100 forums powered by 100 different software packages. Among them, 25 forums were selected for training set and the remaining 75 were kept for testing. These 100 packages cover a wide group of forums.

3.1 Evaluation of FoCUS Models:

To build page classifiers, we manually selected five index pages, five thread pages, and five other pages from each of the 40 forums and extracted the features. For testing, we manually selected 10 index pages, 10 thread pages, and 10 other pages from each of the 160 forums. This is called 10-Page/60 test set. We then ran Index/Thread URL Detection module. We computed the results at page level not at individual URL level since we applied a majority voting procedure. To further check how many annotated pages FoCUS needs to achieve good performance. We conducted similar experiments and applied cross validation. We find that our page classifiers achieved over 96 percent recall and precision at all cases with tight standard deviation. It is particularly encouraging to see that FoCUS can achieve over 98 percent precision and recall in index/thread URL detection with only as few as five annotated forums.

3.2 Evaluation of Page-Flipping URL Detection

To test page-flipping URL detection, we applied the module “Page-Flipping URL Training Set” on the 10-Page/160 test set and manually checked whether it found the correct URLs. The method achieved 99 percent precision and 95 percent recall.

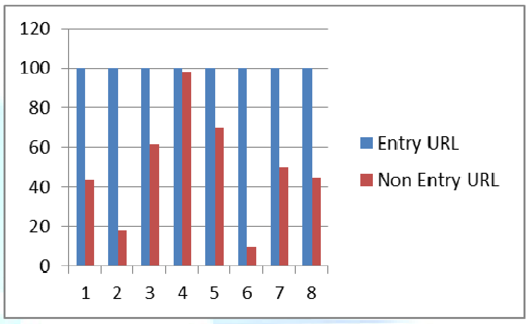

Fig 3 Covering comparison between starting from entry URL and non entry URL

3.3Evaluation of Entry URL Discovery

All prior works in forum crawling assume that an entry URL is given. However, finding forum entry URL is not trivial. To demonstrate this, we compare our entry URL discovery method with a heuristic baseline. For each forum in the test set, we randomly sampled a page and fed it to this module. Then, we manually checked if the output was indeed its entry page. In order to see whether FoCUS and the baseline were robust, we repeated this procedure 10 times with different sample pages. The baseline had 76 percent precision and recall. On the contrary, FoCUS achieved 99 percent precision and 99 percent recall. The low standard deviation also indicates that it is not sensitive to sample pages.

4. Conclusions

The FoCUS, a supervised forum crawler. It reduced the forum crawling problem to a URL type recognition problem and showed how to leverage implicit navigation paths of forums, i.e., EIT path, and designed methods to learn ITF regexes explicitly. Experimental results on 160 forum sites each powered by a different forum software package confirm that Focus can effectively learn knowledge of EIT path from as few as five annotated forums. These learned regexes can be applied directly in online crawling. Training and testing on the basis of the forum package makes our experiments manageable and our results applicable to many forum sites. Focus can start from any page of a forum. The results on nine unseen forums. Results on 160 forums show that Focus can apply the learned knowledge to a large set of unseen forums and still achieve a very good performance. Though the method introduced in this paper is targeted at forum crawling, the implicit EIT-like path also applies to other sites, such as community Q&A sites and blog sites. In future, we like to discover new threads and refresh crawled threads in a timely manner. The initial results of applying a Focus-like crawler to other social media are very promising. We would like to conduct more comprehensive experiments to further verify our approach and improve upon it.

References

[1] Anirban Dasgupta Ravi Kumar Amit Sasturkar"De-duping URLs via Rewrite Rules"2008.

[2] A. Agarwal, H. S. Koppula, K. P. Leela, K. P. Chitrapura,S. Garg, P. K. GM, C. Haty, A. Roy, and A. Sasturkar. Url normalization for de-duplication of web pages 1987–1990,November 2009.

[3] S. Brin and L. Page,"The Anatomy of a Large-Scale Hypertextual Web Search Engine."vol. 30,nos. 1-7, pp. 107-117, 1998.

[4] R. Baumgartner, S. Flesca, and G. Gottlob.Declarative information extraction, Web crawling, and recursive wrapping with Lixto.2173, 2001.

[5] A. Z. Broder, M. Najork, and J. L. Wiener. E_cient URL caching for World Wide Web crawling. In International conference on World Wide Web, 2003.

[6] Z. Bar-Yossef, I. Keidar, and U. Schonfeld. Do not crawl in the DUST: different URLs with similar text. In Proc. 16th WWW, pages 111−120, Banff, Alberta, Canada, May 2007.

[7] S. Chaudhuri, V. Ganti, and R. Motwani. Robust idenfication of fuzzy duplicates. In Proc. 21st ICDE, pages 865–876, 2005.

[8] D. Fetterly, M. Manasse, and M. Najork. Detecting Phrase-Level Duplication on the World Wide Web. To appear in 28th Annual International ACM SIGIR Conference (Aug. 2005).

[9] C. Gao, L. Wang, C.-Y. Lin, and Y.-I. Song,"Finding Question-Answer Pairs from Online Forums,"pp. 467-474, 2008.

[10]N. Glance, M. Hurst, K. Nigam, M. Siegler, R. Stockton, and T. Tomokiyo, "Deriving Marketing Intelligence from Online Discussion,",pp. 419-428, 2005.

[11]H. Garcia-Molina and L. Page. Efficient crawling through URL ordering. in: Proc. of' the 7th Intermitiontrl World Wide Web Conference April l5- IS, 1998.

[12]T. H. Haveliwala, A. Gionis, D. Klein, and P. Indyk. Evaluating strategies for similarity search on the Web.In Proc. 11th International World Wide WebConference, pages 443,442, May 2002.

[13]M. Henzinger,"Finding Near-Duplicate Web Pages: A Large-Scale Evaluation of Algorithms,",pp. 284-291,2006.

[14]H.S. Koppula, K.P. Leela, A. Agarwal, K.P. Chitrapura, S. Garg, and A. Sasturkar,"Learning URL Patterns for Webpage De-Duplication,"Proc. Third ACM Conf. Web Search and Data Mining,pp. 381-390, 2010.

[15]J. Myllymaki. Effective web data extraction with standard XML technologies. In Proc. WWWW10,pages 689–696, May 2001.

[16]Pantel, Patrick., Eric Crestan, Arkady Borkovsky, Ana-Maria Popescu, Vishunu Vyas. 2009. Web-Scale Distributional Similarity and Entity Set Expansion.In Proceedings of. EMNLP, 2009.

[17]U. Schonfeld and N. Shivakumar, "Sitemaps: Above and Beyond the Crawl of Duty,",pp. 991-1000, 2009.

[18] R. Song, H. Liu, J.-R. Wen, W.-Y. Ma. Learning important models for Web page blocks based on layout and content analysis. ACM SIGKDD Explorations Newsletter,6(2):14–23, Dec. 2004.

[19]Y. Wang, J.-M. Yang, W. Lai, R. Cai, L. Zhang, and W.-Y. Ma,"Exploring Traversal Strategy for Web Forum Crawling,",pp. 459-466, 2008.