ISSN:0975-9646

Aleem Ansari et al, / (IJCSIT) International Journal of Computer Science and Information Technologies, Vol. 6 (2) , 2015, 1865-1868

A Novel Page Links Prediction Technique for Web Search Sources

Aleem Ansari[1], Dr. Hemlata Vasishtha[2]

[1]Ph.D. Scholar, [2]Professor, Faculty of Engineering & Technology,
Shri Venkateshwara University, Gajraula, India

***Abstract*— Search results (Data records) retrieved from web sources such as search engines or dynamic websites (e.g. online shopping) are usually scattered among different web pages. Each of these response pages displays fixed number of records ordered by certain search criteria. These response pages usually contain one or more hyperlinks that allows user to navigate to other response pages. Certain applications like web data extraction sometime needs to access only response pages that belong to certain search criteria. However current web crawlers cannot distinguish between related response pages and other pages from the single web source. In this paper we have proposed a simple and effective approach for identifying the URLs of the subsequent response pages from a web search source. Our approach takes the URLs of second and third response pages as input and generates the URLs of remaining pages as output. We have employed Myer's diff algorithm [1] for determining the differences between parameters in the input URLs. After identifying key parameters and their differences we construct URLs for remaining pages by assigning proper weight to key parameters.**

***Keywords*— Web Search Source, Crawler, Data Record Detection, Information Extraction, Myer's diff algorithm, Web Content Mining.**

I. INTRODUCTION

Search results (Data records) retrieved from web sources such as search engines or dynamic websites (e.g. online shopping) are usually scattered among different web pages [2]. Each of these response pages displays fixed number of records ordered by certain search criteria. These response pages usually contain one or more hyperlinks that allows user to navigate to other response pages. Certain web data extraction algorithms as mentioned in as mentioned in [3], [4], [5], [6] and [7] sometime needs to access only response pages that belongs to certain search criteria.

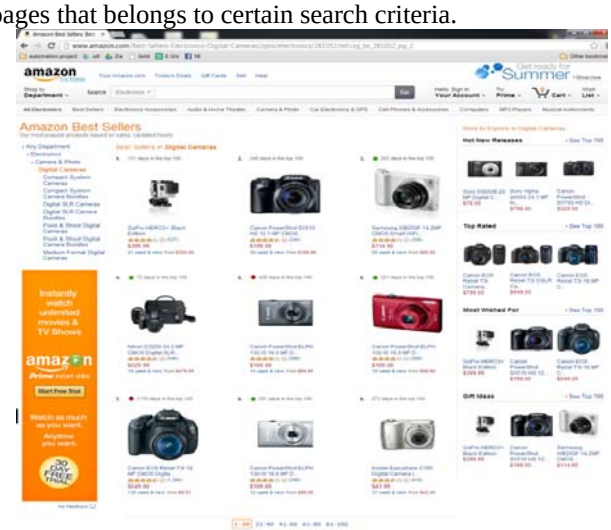

Fig. 1 Sample web pages displaying product information

Figure 1 shows sample response page from Amazon website displaying list of cameras. The web page lists information about nine different cameras. At the bottom of the page we can see navigation links pointing to other response pages. User navigates through these navigation links to view remaining results (cameras) scattered on other web pages. In this paper, we refer to such navigation links as Subsequent Page links (SPLs).

Often different web sources display these SPLs in different format. Figure 2 displays few sample SPL formats used by some of the web sources. After surveying many web sources, in this paper we classify SPLs into two different types:

(a) Multi SPL: In this type, any response page has several SPLs in the form of hyperlinks pointing to several other response pages.

(b) Single SPL: In this type, any response page has only one and or two SPL(s) in the form of hyperlink pointing to previous and or subsequent response page(s) respectively.

Fig. 2 Multi SPL and Single SPL

The above SPLs types can be in the form of links, buttons, images, etc

Web Data Extraction systems often need to access these web pages for data extraction. Eventually, extracted data might be post-processed, converted in the most convenient structured format and stored for providing value added services such as comparative shopping, market intelligence, meta-querying and search ([3], [4], [5], [6], [7]).

The first step in many Web Data Extraction systems is to access the collection of web pages from a web source. However due to the vast nature of web sources it is very difficult to group the web pages according to certain criteria for e.g. separating pages containing mobile listings from laptops listings. Currently there are no available algorithms that can segregate between different categories of web pages accurately.

The main aim of our work is to devise an algorithm that will access/download only pages that belong to certain category listing. The advantage is processing and network bandwidth savings. This also results in elimination of redundant and unrelated web pages thereby aiding in efficient data extraction.

In this paper we have presented highly effective and efficient solution for determining subsequent response pages. We have also implemented this algorithm. The rest of the work is organized as follows. Section 2 describes about the progress being done till yet in this area. Section 3 describes our algorithm in detail. Section 4 lists the results that were found during the testing of the algorithm. Finally section 5 concludes the shortcomings and further work that need to be done for the betterment of the algorithm.

II. Related Work

Web crawlers are relatively automated programs that methodically scan through internet pages to create an index of the data they are looking for. Web crawlers are also known as web spider, web robot, bot, and automatic indexer. When employed in search engines, web crawlers searches the HTML document for given web site in the internet. It stores a copy of page that has been visited by search engines and indexing is performed.

Focused Web Crawling [8] is intended for the people who are interested in a small fraction of the Web. The Focused crawlers are used to discover the set of pages that covers certain topic. Focused Web Crawling helps in time, network bandwidth and storage savings.

A. Fetching Web data

Web data integration systems often need to extract data from each of its web sources distributed over the Web. In a favourable environment, an integration system may have special agreement with its web sources on how to transfer data or utilizing Web Services. If the number of web sources are less and stable, an integration system could have customized programs for different web sources to fetch data. However, when an integration system needs to manage hundreds of web sources, especially when they are autonomous and heterogeneous and changes their interfaces frequently in unpredictable way; highly automated, adaptive, and robust methods are needed for the integration of the fetched data.

For automatic processing, the integration system should have the capability to automatically discover search interfaces, send queries and fetch the response pages. However, it is not clear how an integrator can fetch data from subsequent response pages. In this paper, we assume that we are able to connect to a Web search source's interface, submit required queries, and fetch the first response page returned.

B. Wrapper Generations

Wrapper generation is a data extraction enabling process that has been employed extensively in the area of Structured Data Extraction [10], [11]. If we can identify SPLs and fetch the corresponding pages, wrappers can be generated to process these pages to extract the records contained in them. In the context of this paper, a wrapper is used to extract a specified response page from a Web search source rather than the result records contained in a particular response page.

III. Subsequent Response Page Prediction

We first define few terms that we use for the purpose of simplifying the description of our approach:

Definition 1: Incorrect Query: When a search query q is submitted to a Web search source S, resultant records R are scattered into set of P pages with each page p containing C resultant records (such that $C>0$). Query q is an Incorrect Query to S when $R = 0$ and $P = 0$

Definition 2: Optimal Query: When a search query q is submitted to a Web search source S, resultant records R are scattered into set of P pages with each page p containing C resultant records (such that $C>0$). Query q is an Optimal Query to S when $R > C$ and $P > 1$.

Definition 3: Correct Response Page: The response pages P returned for optimal query is called as Correct Response Page (*CRP*).

Definition 4: Incorrect Response Pages: The response pages P returned for an incorrect query is called as Incorrect Response Page (*IRP*).

The proposed algorithm takes URLs of second and third pages of a Web search source as input and returns list of candidate SPLs L such that each SPL in L maps to one of the URLs of pages in P.

A. *Probe Query Generation*

First, we need to generate an optimal query corresponding to the input Web source. In our running example, 'datamining' is the optimal query *qi* to web source google.com. Thus the URLs (SPLs) for the first three pages are as follows:

https://www.google.com/search?q=datamining
https://www.google.com/search?q=datamining&start=10
https://www.google.com/search?q=datamining&start=20

B. *Observations*

Generally data extraction algorithms like [2], [3], [4], [5] and [12] assumes that the data to be extracted from the Web pages follows few regularities when displayed (e.g., similar layout, similar record structure, similar tag strings, etc.). Similarly, we can observe that URLs of SPLs generated by a Web source in response to a query also follow some regularity. The reason is that these response pages are generated by dynamic programs on the server side whose results depend on the supplied query parameters.

When we submit a query to a web search source, the query appears in the URLs of the results as name value pair. The name value pair with query as its value is called as query parameter. For example when we submit Search another query say 'webmining' to Google we get URLs(or SPLs) for the first three pages as follows:

https://www.google.com/search?q= webmining
https://www.google.com/search?q= webmining &start=10
https://www.google.com/search?q= webmining &start=20

In above cases we can see that the search query appears in the URLs (SPLs) with query parameter q having value as webmining.

In every SPL we have at least one parameter that is used to distinguish SPLs from one another and hence subsequent pages. Such parameters are called page parameter(s).The two SPLs for a single query to a web

search source only differ in page parameter(s). If we ignore page parameter(s), then all the SPLs will be the same. Thus by appropriately changing the values of page parameter(s), we can fetch the desired response pages from the web search source. Hence the main goal is to identify page parameters in the SPLs and difference δ between two successive pages parameter(s). Once identified, we simply increment the page parameter(s) by the difference δ to fetch the required page. For previous example we have delta =10

Thus after knowing the page parameter for second and third pages *p2* and *p3* respectively we can fetch the page number *p* using the formula:

https://www.google.com/search?q=webmining&start=*S* such that *S=((p2 + (p3 – p2))*p).*

Thus after constructing SPLs for required pages we can easily fetch them for data extraction or similar purpose.

C. *Pseudo code for Page Parameter Normalization*

1) Input URLs of second and third page say *URL2* and *URL3*.
2) Find list of differences between the *URL2* and *URL3* using Myer's diff algorithm. where each item in the list is in the one of the following format:
   =(str) means string *str* is common in both the URLs.
   -(str) means string *str* is deleted from the first URL i.e. *URL2*.
   +(str) means string *str* is added to the first URL i.e. *URL2*.
3) If *str* is deleted from *URL2*, get the position of *str* in *URL2*
4) Find the digits that are near *str* (both left side and right side) in *URL2*. Let's call this *strDigit2*.
5) Find the corresponding digits that are near position of *str* (both left side and right side) in *URL3*. Let's call this *strDigit3*.
6) Replace *strDigit2* and *strDigit3* with a unique string every time say *myparam1*, *myparam2*, *myparam3* and so on.
7) Save data in the format *myparam1*, *strDigit2*, *strDigit3* in the list say *paralist*.
8) Repeat through step 2 until there is no difference between *URL2* and *URL3*.

Once we normalize the page parameters using the page parameter normalization technique, both URL2 and URL3 becomes identical. The next step is to predict SPL for required page p by appropriately replacing page parameter values.

D. *Pseudo code for Subsequent Page Link Prediction*

1) To get the URL for page number *p*, create copy of the *URL2* say *newURL*.
2) Get the first item from the list *paralist.*
3) Get parameter name say *myparam1*, *strDigit2* and *strDigit3* from the list item.
4) Replace *myparam1* in the *newURL* with *(strDigit2 + (strDigit3 – strDigit2))*p*.
5) Repeat from step 2 with remaining items in the list i.e. *myparam2*, *myparam3* and so on.
6) Now the *newURL* is the final URL for page no *p*.

IV. EXPERIMENT

The proposed work is implemented using java. To analyse the validity and performance of our algorithm we made it to pass through various set of web pages. During our experiments we came across response web pages of different categories.

A. *Response page with no SPL*

In this category the web search source returns few records for viewing purpose. The remaining records are displayed dynamically using AJAX or JavaScript based on user's interactions on the same page. Our technique is not applicable to this category since there is no SPL.

B. *Response page with Numeric Page Parameter(s)*

In this category more than one response pages are returned by web search source. Each response page contains SPL for navigating to other pages. However the page parameters have numeric values. Our technique achieves 100% accuracy for response pages in this category.

C. *Response page with Non-Numeric Page Parameter(s)*

In this category the more than one response pages are returned by web search source. Each response page contains SPL for navigating to other pages. However the page parameters have non-numeric values. Our technique is not applicable to this category since it cannot identify the difference between the page parameters accurately.

D. *Execution Time*

Execution time: Average time taken for our approach is *O(1)* to fetch the required page. Once a data structure is created for a Web source after page parameter normalization, any specified response page can be fetched from that source in a fraction of a second.

V. CONCLUSIONS AND FUTURE WORK

In this paper we proposed a simple yet novel technique for automatically fetching desired response page from Web search source. It is a vital task for information integration systems since search results retrieved from web sources are usually scattered among different web pages. The proposed first normalizes the page links using Myer's diff algorithm and then fetches the required response pages. The algorithm works only for SPLs with numeric page parameters. Since most of the websites use SPLs with numeric page parameters this algorithm provide great benefits to various Knowledge Discovery Applications such as comparative study of products or services from various companies, smart shopping, etc.

We also observed the scenarios of page parameters with non-numeric values. Our technique is not applicable to this category since it cannot identify the difference between the page parameters accurately The future work would be normalization of page links with non-numeric page parameter(s).Finally testing algorithm on more and more web sources will be continuous process to fine tune it for any corner cases.

REFERENCES

[1] Myers, Eugene W. "AnO (ND) difference algorithm and its variations." Algorithmica 1.1-4 (1986): 251-266.

[2] Novotny, Róbert, Peter Vojtas, and Dušan Maruscak. "Information Extraction from Web Pages." Proceedings of the 2009 IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology-Volume 03. IEEE Computer Society, 2009.

[3] Laender, Alberto HF, et al. "A brief survey of web data extraction tools." ACM Sigmod Record 31.2 (2002): 84-93.

[4] Liu, Bing, and Yanhong Zhai. "NET–a system for extracting web data from flat and nested data records." Web Information Systems Engineering–WISE 2005. Springer Berlin Heidelberg, 2005. 487-495.

[5] Ye, Shiren, and T-S. Chua. "Learning object models from semistructured web documents." Knowledge and Data Engineering, IEEE Transactions on 18.3 (2006): 334-349.

[6] Baumgartner, Robert, Georg Gottlob, and Marcus Herzog. "Scalable web data extraction for online market intelligence." Proceedings of the VLDB Endowment 2.2 (2009): 512-1523.

[7] Kayed, Mohammed, and Chia Hui Chang. "FiVaTech: Page-level web data extraction from template pages." Knowledge and Data Engineering, IEEE Transactions on 22.2 (2010): 249-263.

[8] Chakrabarti, Soumen, Martin Van den Berg, and Byron Dom. "Focused crawling: a new approach to topic-specific Web resource discovery." Computer Networks 31.11 (1999): 1623-1640.

[9] Liu, Hongyu, and Evangelos Milios. " Probabilistic Models for Focussed Web Crawling." Computational Intelligence 28.3 (2012): 289-328.

[10] Chakrabarti, S . Mining the Web: Discovering Knowledge from Hypertext Data. Morgan Kaufmann Publishers, 2002.

[11] Chang, Chia Hui, et al. "A survey of web information extraction systems." Knowledge and Data Engineering, IEEE Transactions on 18.10 (2006): 1411-1428.

[12] Zhai, Yanhong, and Bing Liu. "Web data extraction based on partial tree alignment." Proceedings of the 14th international conference on World Wide Web. ACM, 2005.