TECHNICAL TRANSACTIONS | **CZASOPISMO TECHNICZNE**
MECHANICS | MECHANIKA

1-M/2013

ANDRZEJ OPALIŃSKI*, WOJCIECH TUREK*, MIROSŁAW GŁOWACKI*

INFORMATION MONITORING BASED ON WEB RESOURCES

MONITORING INFORMACJI W OPARCIU O ZASOBY SIECI WEB

Abstract

The paper summarizes the system for WEB resources monitoring based on defined query. Experiment compares results returned by the proposed system to those provided by Google Search and Google Alert services. Results indicate that the system could be solid base for development and tests of pattern detection and information retrieval mechanism, while providing more data than Google solutions. Drawback of system and further development plans are also presented.

Keywords: *crawling*, *WEB monitoring*, *information retrieval*

Streszczenie

W artykule przedstawiono architekturę systemu monitorującego zasoby sieci WEB pod kątem zdefiniowanego zapytania. Wyniki działania systemu porównano z prowadzonym w tym samym czasie monitoringiem za pomocą mechanizmów oferowanych przez Google. Rezultaty wskazują, że system może być przydatną bazą do badania mechanizmów wykrywania wzorców i wyszukiwania informacji, udostępniając więcej danych w porównaniu do mechanizmów Googla. Wykazano też niedoskonałości aktualnej wersji systemu wynikające ze specyfiki źródeł danych i zaproponowano kierunki jego rozwoju.

Słowa kluczowe: *crawling*, *monitoring Internetu*, *wyszukiwanie informacji*

* MSc. Eng. Andrzej Opaliński, PhD. Eng. Wojciech Turek, Prof. Eng. Mirosław Głowacki, Department of Applied Computer Science and Modelling, Faculty of Metals Engineering and Industrial Computer Science, AGH University of Science and Technology.

1. Introduction

The growth of the World Wide Web, which has been observed over last years, has resulted in the greatest base of electronic data. It is even hard to estimate real size of the Web. The WorldWideWebSize.com portal claims that the most popular search services index more than 50 billion Web pages [1]. In 2008 Google published an information, that the indexer found 1 trillion unique addresses [2]. These estimates definitely do not show the real size of the Web because the indexers deliberately ignore particular fragments, like content generators, link farms or pages with illegal content.

The features of the Web pose huge challenges for searching systems. The size itself creates significant scalability and performance issues. What is more, it is very hard to acquire information about a content which is really looked for by an user and detect pages containing information needed by an user.

WEB crawling in information retrieval domain is well described issue, spread along with the Internet development. Numerous summarized surveys were made in this topic [3,4]. Many researchers described information retrieval related problems, and presented their own solutions to deal with it. Pandey used agent-based system to solve the problem of crawling order [5], Manku et.al. focused on finding and eliminating duplicates in crawl process [6], and Broder et.al. proposed efficient mechanism for url caching in this research area [7].

Mostly due to the scale problem, there are still many unresolved aspects in this area. Publicly available Web search services offer access to very simple and fast ways of finding pages. This method of finding information in the Web is used every day by each Internet user. However, several significant drawbacks and limitations of the approach do exist. Firstly, if several pages contain all specified words, ordering of results is imposed by the search engine. Sorting is typically based on webpage's popularity. This feature connected with the limit in the number of found pages, results in inability of finding some pages. Secondly, the query language is typically very simple. It is impossible to express advanced patters concerning sentences or use of synonyms. It is even impossible to describe clearly the rules specifying letters casing, distance between words or words ordering. Thirdly, low frequency of crawling causes outdated results. The searchers often find pages which contain different content than expected or no longer existing one.

These limitations encourage researchers to continue work on different ways of finding valuable information in the Web. The subject of focused crawling has received significant attention over the last few years. The idea of a crawler which can select pages relevant to a specified topic [8] has been implemented using various techniques [9, 10]. Most obvious application of a focused crawler is a topic-specific search service, which can provide more accurate results.

Use of index-based search engines can successfully direct a user to potentially interesting Web sites. However, when the content of the Web sites changes fast and the information must be detected as soon as possible after it is published, indexing-based methods becomes insufficient. When a user knows where to look for results, but it is impossible to watch the Web sites continuously, a different approach to the problem of searching the Web is needed.

A lot of proprietary tools is offered for WEB crawling purposes. Even Microsoft released this kind of software, called FAST Search Web crawler, as a part of Share Point Server 2010

[11]. There are also a few open sources solutions, compared by Girardi [12]. One of the most popular is Heritix [13] and Websphinx [14]. Those tools allows to crawl and collect data from specified domains, but scalability and more advanced modifications are the drawbacks. Many companies offers such services, returning just a results, based on requested query. Of course such a services are not free of charge, and providers does not present an algorithm of their solutions. There is also one common free service of this type – Google Alert [15]. It offers query monitoring and returns results as a link sent by an email.

In this paper, a system for crawling and monitoring selected fragments of the Web is presented. It provides a service, which can monitor precisely specified fragments of the Web and actively report when a particular pattern is found in newly-published content. Possible applications of the system include monitoring auctions services or job advertisements. It can also be used by law enforcement services for detecting illegal content quickly.

Monitoring of Polish Internet resources for query "shale gas energetics" is presented as an experiment. Results returned by proposed system are compared to the results provided by Google Search Engine and Google Alert service for the same time and query.

2. The architecture of the presented system

The architecture of the system is inspired by a Java-based general purpose Web crawler with indexer presented in [16, 17]. The system can be deployed in three different ways. The less hardware-expensive version is to deploy the system on single PC computer, where all components are running on the machine. This configuration uses particular settings, that significantly limit required system resources and can be used only for monitoring several small Web domains. The second version is a sever based deployment, where a MySQL database server and an JBoss application server are executed on a powerful machine. In this configuration several users can use the same server. Tests showed that a single server can process around 100 000 Web pages every hour. The most robust configuration is based on cluster of servers. It could be used to monitor large data sources, because the performance of processing in this configuration can be easily increased by adding new servers to the cluster.

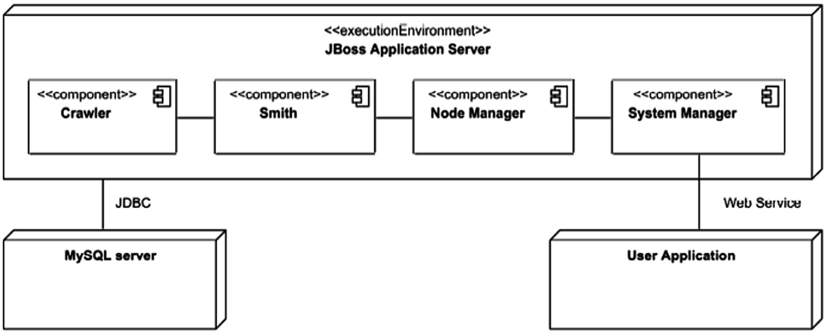

Fig. 1. Abstract architecture of the Web Monitoring System

Rys. 1. Architektura systemu monitoringu informacji w oparciu o dane z sieci WEB

The abstract architecture of the system, which consists all main components is presented in Figure 1. The Crawler component is a single processing thread. It contains a queue of URLs to download and analyze. It is responsible for performing all operations needed to process a Web Page – details on processing algorithms will be presented in the next section. The most important result of the processing is URLs detection – the URLs are returned to the Smith Component. The Client application uses provided Web Service interface. While running configured on cluster of servers The Smith component controls multiple Crawler threads. It starts specified number of Crawlers, manages URLs queues, receives found URLs and communicates with the Node Manager. When running in cluster-based configuration each node used by a system has a single Node Manager which is responsible for communication with global System Manager. The System Manager is responsible for controlling nodes. It collects and distributes found URLs, performs distributed search and provides access to management interface of every node. It also provides a Web Service interface for clients of the system.

3. Resources Processing Algorithm

The most important part of the system is implemented by the Crawler component. It performs processing of Web pages content downloaded from the Internet. A diagram of steps performed by the Crawler is shown in Figure 2. The process of crawling is controlled by the Manager, which keeps a queue of URLs to process. All URLs found by the node are stored in the Urls database. The Manager continuously executes the processing sequence, that consists of the following steps:

- Resource downloading, which results in HTML source stored in a memory buffer,
- HTML parsing by the Lexer and the document model building,
- Changes detection algorithm, avoid storing already processed webpages,
- Content processing by various plugins.

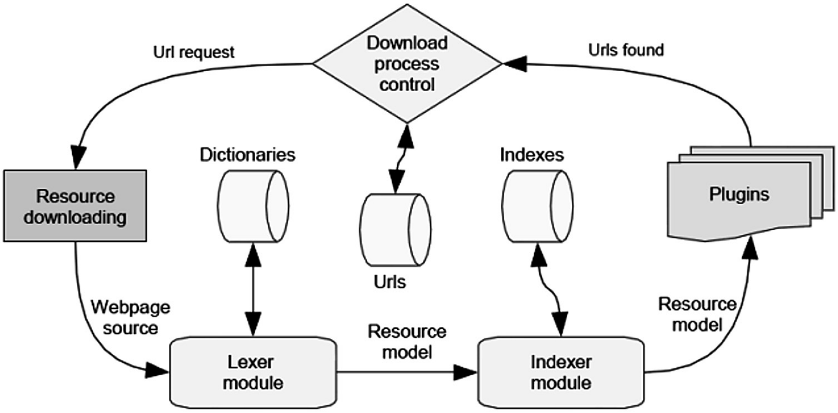

Fig. 2. Processing performed by a single crawler component

Rys. 2. Proces przetwarzania pojedynczego crawlera

This sequence is being executed by every single URL which appears on the list of the Manager. The following sections provide more details on the processing algorithms.

3.1. Parsing and Resource Model Building

The Lexer converts HTML source into a resource model. The model represents a tree structure built of segments. Each segment represents a selected structural element of a Web page (tables, paragraphs and lists). Segment can contains other segments, or can be leaf-segment, containing lists of words, special characters and HTML tags. Each element of the HTML source is converted to an element of the tree structure or to a token. There are three basic types of tokens: words, tags and special characters and each token has its unique identifier. The content of each leaf segment is converted into a list of identifiers, making following processing very efficient.

The dictionary of words is a very large data structure. Average Web page contains several thousand words, however typically very few are new words. Nevertheless, the size of the words dictionary can reach millions of entries after a few days of crawling. Therefore the implementation uses large in-memory caches based on hash maps to make the word-to-identifier conversion as fast as possible.

3.2. Changes Detection Algorithm and Content Processing

The changes detection algorithm is based on hash codes calculated for analyzed content. The hash code for a segment containing is calculated using tokens' ids based on the idea used by Java String class implementation. The hash codes are calculated for every leaf segment. If the hash code has been found in any previous processing of the same Web page, the segment is considered unchanged and is not processed any further.

To determine what values of hash codes have been already processed, the Content cache database is used. It stores all hash codes of leaf segments found in the page content. Leaf segments that are considered new or modified, are processed by all enabled plugins.

There is one plugin which is mandatory for proper functioning of the system. The URL detector plugin must be enabled for continuation crawling process. It finds URLs in the content of provided segments, searching for anchor HTML tags. The pattern detection plugin used in the presented version of the system is based on the list of optional words, and required match threshold. A segment will match the pattern if the number of specified words found in the segment exceeds the threshold. This plugin can also use stems instead of words. Other efficient methods for patterns detection could be applied by implementing further plugins. This mechanism provide much more flexibility than the query languages provided by the most popular publicly available search services.

4. Experiments and evaluation

To test the presented system an experiment was performed. The monitored query was "shale gas energetics" and it was applied to the Google Search and and Google Alert services. As a source for the proposed system, there were selected 7 websites of the Google Results list, with highest rank by Alexa web metric [18]. It tends to be the most popular and dynamically changing its content, while it's expected to be the best source for an experiment. Results of the test, which lasted for 70 hours are presented in Table 1.

Table 1

Number of results returned by systems

Domains	Presented system				Google				
	distinct urls	urls with content changes	content related	content related after 9 hours	Google Search	"site:"	"site:"+ duplic.	total count	Google Alert
1	680	1001	47	26	43	564	601	2010	2
2	619	4072	1165	871	11	523	603	3300	2
3	3402	4748	82	28	10	408	607	9660	0
4	3414	4411	21	11	7	632	684	37100	0
5	9268	24556	343	186	11	612	650	10500	1
6	59	82	1	0	6	305	333	1900	0
7	1	1	0	0	4	77	86	77	0

Domains: 1 – serwisy.gazetaprawna.pl, 2 – gazownictwo.wnp.pl, 3 – wyborcza.biz, 4 – forsal.pl, 5 – cire.pl, 6 – egospodarka.pl, 7 – energetyka.pl

Selected domains were crawled by presented system every 6 hours, with the average single crawl time about 30 minutes. Match pattern threshold was set to 2 of 3 words. Results of total crawl period for exemplary domain are presented in Figure 3. Over 90% of results are collected after first crawl process. A logarithmic scale has to be apllied to present the results. Table 1 contains results corresponding to each domain. Values in columns represents: 1 – domain ID, 2 – total number of distinct URLs containing pattern, 3 – total number of URLs, including changing content, 4 – number of webpages with HTML title related to the query, 5 – number of URLs including results after 9th hour of the test – that is comparable to the Google Alert service results. Values in columns corresponding to Google part of table are: 7 – number of urls from domain returned by default Google Search, 8 – number of results with "domain:" option set (results just from specified domain), 9 – "domain:" option including duplicates, 10 – declared number of results, 11 – number of results returned by Google Alert service.

There are some interesting observations, that could be concluded by examination of those results. At first, the number of results returned by default Google Search is just a minor part (1–7%) of results from entire domain – obtained by "site". Most default Google Search results are links from static domains – only 15% comes from news portals, and only 11% comes from domain that are later sources for Google Alerts. Google Search engine has never returned more than 684 result links. It reported up to 37100 total results, but did not returned links to the remaining part of results. Google Alert service returned just 6 results during the experiment duration. It correspond to more than 1000 results returned by the system presented in this article. Vast number of results is an effect of trivial content changes algorithm, vulnerable to a dynamic add-keyword-tags systems, which are applied with common keywords, that was a part of our query.

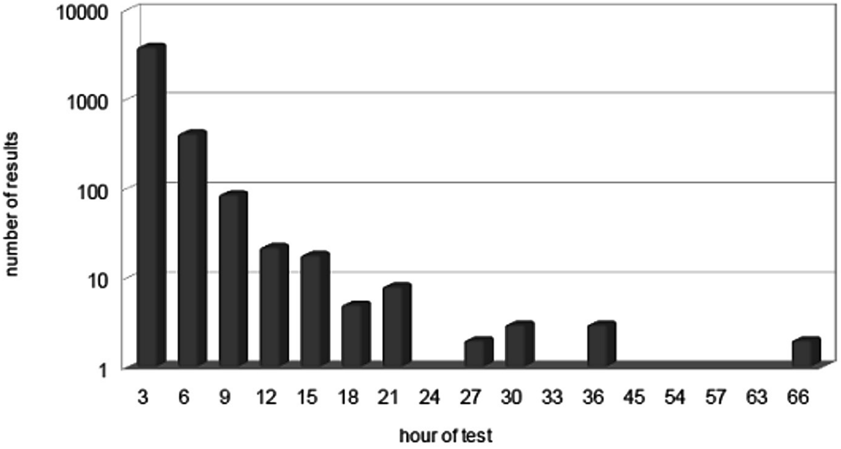

Fig. 3. Results number for forsal.pl domain

Rys. 3. Liczba rezultatów w dla domeny forsal.pl

5. Conclusions and Further Work

The results of the tests highlighted presented system weaknesses and benefits. First is the consequence of trivial content change detection algorithm – high rate of false positive results number, based on tag-keywords html segments. It is very susceptible applied to popular keywords. Advanced methods of content meaning recognition are also required for application in information retrieval domain. Although the benefits of proposed architecture are undeniable. It provides convenient base to testing and development algorithms for pattern detection and information retrieval. It could be adopted to particular user needs, by specifying advanced detection algorithm and provide large amount of data for further processing. It grants an independence comparing to the usage of Google services, where user has very limited influence of returned results. Also scalability is a great advantage, as the system could be configured to be launch as low-cost or cluster-based configuration.

Where Google services are useful in simple and general use cases, presented system gives the potential to by much more flexible and adaptable in more advanced purposes.

Further research should account implementing advanced techniques of subject detection and dynamic tagging independence. Also methods for defining advanced content patterns should be developed. The possibility of defining semantic meaning of the content or similarity to a given text, rather than specifying a list of words, would be very useful.

References

[1] Kunder M., *WorldWideWebSize.com*, 12.2012.

[2] Alpert J., Hajaj N., *We knew the web was big...* (http://tinyurl.com/crzays7–25.07.2008).

[3] Croft, W.B., Metzler D., Strohman T., *Search engines: Information retrieval in practice*, Addison-Wesley 2010.

[4] Kobayashi M., Takeda K., *Information retrieval on the web*, ACM Computing Surveys (CSUR), 32(2), 2000, 144-173.

[5] Pandey S.K., Mishra R.B., *Intelligent Web mining model to enhance knowledge discovery on the Web*, In Parallel and Distributed Computing, Applications and Technologies, 2006. PDCAT'06. Seventh International Conference on, 339-343, IEEE.

[6] Manku G.S., Jain A. & Das Sarma A., *Detecting near-duplicates for web crawling*, In Proceedings of the 16th international conference on WWW, 141-150, ACM, 2007.

[7] Broder A.Z., Najork M., Wiener J.L., *Efficient URL caching for world wide web crawling*, In Proc. of the 12th international conf. on WWW, 679-689, ACM, 2003.

[8] Menczer F., Belew R.K., *Adaptive Information Agents in Distributed Textual Environments*, Proc. of the 2nd Int. Conf. on Autonomous Agents, ACM, 1998, 157-164.

[9] Dong H., Hussain F.K., Chang E., *State of the Art in Semantic Focused Crawlers*, Computational Science and Its Applications, Seoul, Korea, 2009, 910-924.

[10] Dorosz K., Korzycki M., *Latent Semantic Analysis Evaluation of Conceptual Dependency Driven Focused Crawling*, Multimedia Communications, Services and Security, 5th International Conference, MCSS 2012, Krakow 2012, 77-84.

[11] *Crawling Web content with the FAST Search Web crawler*, MS SharePoint library (http://technet.microsoft.com/en-us/library/ff383271%28v=office.14%29.aspx).

[12] Mohr G., Stack M., Rnitovic I., Avery D., Kimpton M., *Introduction to heritrix*, In 4th International Web Archiving Workshop, 2004.

[13] Miller R., *Websphinx, a personal, customizable web crawler* (http://www. cs. cmu. edu/~ rcm/websphinx – 2011-02-12).

[14] Girardi C., Ricca, F., Tonella, P., *Web crawlers compared*, International Journal of Web Information Systems, 2(2), 2006, 85-94.

[15] *Getting Started Guide – What are Google Alerts?* (http://tinyurl.com/csr4z3b).

[16] Turek W., Opaliński A., Kisiel-Dorohinicki M., *Extensible Web Crawler – Towards Multimedia Material Analysis*, Multimedia Communications, Services and Security, 4th International Conference, MCSS 2011, Krakow 2011, 183-190.

[17] Wilaszek K., Wójcik T., Opaliński A., Turek W., *Internet Identity Analysis and Similarities Detection*, MCSS 2012, Krakow 2012, 369-379.

[18] Alexa – provider of global web metrics (http://www.alexa.com – 01.2013).