Proceedings of the 2013 Federated Conference on
Computer Science and Information Systems pp. 1261–1267

Scalable Web Monitoring System

Andrzej Opalinski, Wojciech Turek and Krzysztof Cetnarowicz
AGH University of Science and Technology
Krakow, Poland
Emails: andrzej.opalinski@agh.edu.pl, wojciech.turek@agh.edu.pl, cetnar@agh.edu.pl

***Abstract*—Publicly available Web search engines suffer from several limitations, which significantly reduce usability in particular cases. The most important limitations are out-of-date information, very simple query language and limited number of results. In many cases, users of the Internet are interested in finding new information which appear in the particular Web portal. In this paper, a system for monitoring of Web sites is presented. The system can continuously analyze the content of specified Web pages using advanced text processing algorithms. It actively notifies the user when required information is found in newly-added content. It can be deployed on a single PC as well as on a cluster of computers, providing good scalability. The paper presents an abstract architecture of the system, details of the implementation and real-life experiments results.**

I. Introduction and related works

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 hard to even 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]. Four years ago Google published an information, that the indexer found 1 trillion unique addresses [3]. 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 what a user is really looking for and detect pages containing information needed by the user.

Publicly available Web search services offer access to very simple and fast ways of finding pages. The services use Web crawlers to visit as many pages as possible and build an inverted index of all processed content. The indexes make it possible to find Web pages which contain specified words in few milliseconds. This method of finding information in the Web is used every day by each Web user. However, several significant drawbacks and limitations of the approach do exist:

- If several pages contain all specified words, ordering of results is imposed by the search engine. Sorting is typically based on popularity. This feature connected with the limit in the number of found pages results in inability of finding some pages.
- The query language is typically very simple. It is impossible to express advanced patterns concerning sentences or use synonyms. It is even impossible to specify rules specifying letters casing, distance between words or words ordering.
- Low frequency of crawling causes outdated results. The searchers often find pages which contain different content or no longer exist.
- High popularity of search engines results in an interesting feature of the Web: if a page cannot be found using search services it is considered nonexistent.

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 last few years. The idea of a crawler which can select pages relevant to a specified topic [7] has been implemented using various techniques [8], [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.

Some complex solutions in this area were also presented. Liu et al. [4][5] proposed a system that monitors web resources and reports changes of web content by sending messages for system's users. Although, those solution does not focus on various methods of information pattern detection, as it's presented in this article. The another system - WebMon [9] - is also a tool for web information monitoring, and could be applied to monitor date, keywords or links. It's intended for multiuser access and provides a useful functionalities, but it also doesn't support various pattern detection mechanism. There are also publicly available solutions as Corona tool [6], which is easily scalable decentralized system available for multiple subscribers, but it's detection pattern flexibility is limited.

In this paper a system for 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 a newly-published content. The various pattern detection methods were tested and compared. Also crawl performance and pattern detection is

978-1-4673-4471-5/$25.00

tested and presented, in comparison to standard Google search results. Proposed system could be deployed on a PC, server or cluster - based architecture, to fulfill required performance. 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.

II. Architecture of the Web Monitoring System

The architecture of the system is inspired by a Java-based general purpose Web crawler with indexer presented in [11], [12]. The crawler uses a cluster of computers for parallel processing of different Web sites. It provides a distributed inverted index of all words found on visited pages. The general architecture of the Web monitoring system is presented in a 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 Smith component controls multiple Crawler threads. It starts specified number of Crawlers, manages URLs queues, receives found URLs and communicates with the Node Manager.

Each node used by a system has a single Node Manager is responsible for communication with global System Manager. Each Node Manager provides administration interface for monitoring and management. It also provides a service interface, which is used for executing search queries.

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.

The Client application uses provided Web Service interface. It is implemented in a different technology and provides convenient graphical user interface.

The system can be deployed in three different ways:

1) PC-based,
2) server-based,
3) cluster-based.

The smallest configuration can be executed on a single modern PC. In this configuration all components are running on the same computer. This configuration uses particular settings, which significantly limit required system resources. It does not need expensive hardware however, it can be used only for monitoring several small Web sites.

In the configuration using a single server, the MySQL database server and the JBoss application server are executed on a powerful machine, which is working constantly. User application can be started from time to time in order to verify searching progress. 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 advanced configuration uses a cluster of servers. The performance of processing in this configuration can be easily increased by adding new servers to the cluster.

III. 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 a Figure 2.

The process of crawling is controlled by the Manager, which stores 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, which consists of the following steps:

- Resource downloading, which results in HTML source stored in a memory buffer. This step includes filtering unsupported file formats, HTTP servers error handling and maximum source length verification.
- HTML parsing by the Lexer. This is the most complex and time-consuming step of the processing which builds document model.
- Changes detection, which results in selecting fragments of a Web page content that have never been processed.
- Content processing by various plugins operating on the document model created by the Lexer. One of the plugins returns a list of URLs found in the processed content, that are added to the queue of the Manager.

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.

A. Content Parsing and Resource Model Building

The Lexer converts the HTML source into a resource model. The model represents a tree structure built of segments. Each segment represents a selected structural element of the Web page (tables, paragraphs and lists). Segments can contain other segments or they can constitute leafs of the tree 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:

1) words,
2) tags,
3) special characters.

Each token has its unique identifier – an eight byte integer. Selected ranges of the identifiers are reserved for tags and special characters. The rest is being dynamically assigned to new words found in the content. This approach converts the content of each leaf segment 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

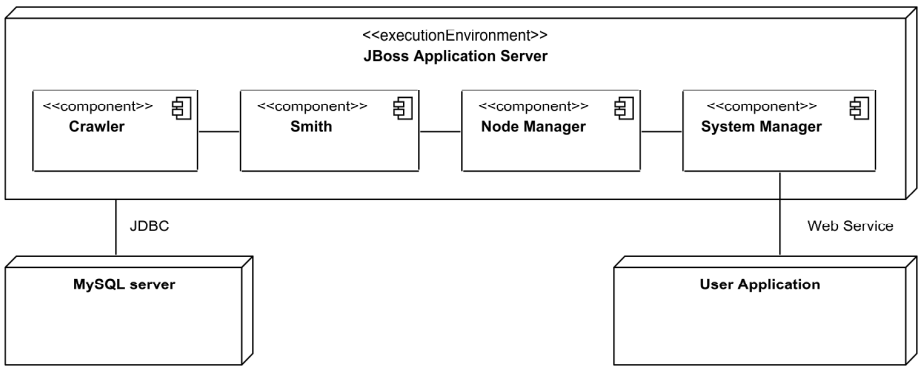

Fig. 1. Abstract architecture of the Web Monitoring System

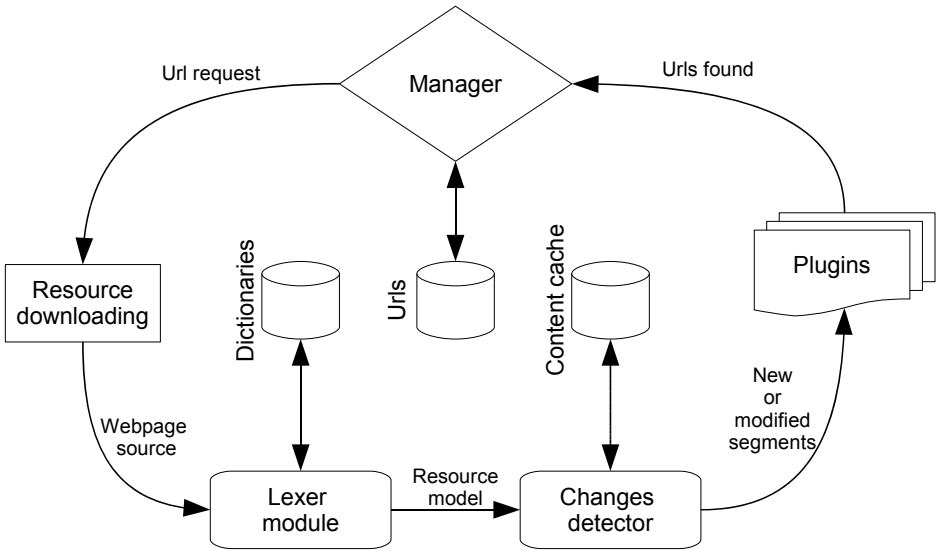

Fig. 2. Processing performed by a single Crawler component.

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.

B. Changes Detection Algorithm

The changes detection algorithm is based on hash codes calculated for analyzed content. The hash code for a segment seg containing n tokens is calculated using tokens' ids in the following way:

$$hash(seg) = \sum_{i=0}^{n-1} seg[i] \cdot 31^{n-1-i} \quad (1)$$

where $seg[i]$ is the identifier of the i^{th} token in the segment. The algorithm is very similar to the one 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. Theoretically, two different segments could have the same hash code, however, using 64 bit identifiers and 64 bit hash codes makes it almost impossible.

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 a page content. Typical Web page contains between 10 and 100 leaf segments.

C. Content Processing

Leaf segments that are considered new or modified, are processed by all enabled plugins. A plugin is a component which provides a common interface – it accepts resource model or its parts.

There is one plugin which is mandatory for proper functioning of the system. The URL detector plugin must be enabled to continue crawling process. It finds URLs in the content of provided segments, searching for anchor HTML tags.

The Web Monitoring System provides several other plug-ins that are used for finding Web pages containing patterns specified by a user. The plugins provide several methods for defining the patterns.

a) List of words: – a user provides a list of words in particular form. A segment will match the pattern if all words are present in the segment.

b) List of stems: – a user provides a list of words in any form. A segment will match the pattern if any form of each word is present in the segment. To implement the functionality a plugin uses a stemmer component which can convert any word to its basic form and provide all possible forms for a given word. The stemmer typically requires a language-specific dictionary of all words and possible forms.

c) List of close words: – a user provides a list of words and maximum distance (in words) between all the words. A segment will match the pattern if all words are present in the segment and the distance between the most distant words is less than the value provided. The order of words in the list is ignored. This plugin can also use stems instead of words.

d) List of words in a sentence: – a user provides a list of words. A segment will match the pattern if all words are present in a single sentence in analyzed the segment. This plugin can also use stems instead of words.

e) List of optional words: – a user provides a list of words and required 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.

The plugins provide several convenient ways of defining precise patterns which a user is looking for. They provide much more flexibility than the query languages provided by the most popular publicly available search services. Particular examples of the patterns and found content will be provided in the next section.

IV. Tests

Special set of tests of the system has been performed after implementing proposed solutions. For testing purposes, four most popular news portals, according to Alexa [2] ranking, were selected :

- interia.pl [1],
- gazeta.pl [2],
- onet.pl [3],
- wp.pl [4].

For the crawl process, five detectors based on the methods described in previous chapter were configured. All of detectors searched for patterns in Polish language, due the fact that in the implementation of the stemmer algorithm and its database was available only for Polish language. It could be easily adapted to other languages by implementing the stemmer algorithm and its database in other languages.

[1]www.fakty.interia.pl
[2]www.wiadomosci.gazeta.pl
[3]www.wiadomosci.onet.pl
[4]www.wiadomosci.wp.pl

Detectors that were used to search for results, are:

- *Simple* – based on *List of words* method, parametrized by words ”virus” and ”flu”,
- *Stem* – based on *List of stems* method, parametrized by words ”virus” and ”flu” and it’s stems,
- *Distance* – based on *List of close words* method, parametrized by words ”virus” and ”flu” and it’s stems and distance between first and last word equal 5,
- *InPhrase* – based on *List of words in sentence* method, parametrized by words ”virus” and ”flu” and it’s stems,
- *Percentage* – based on *List of optional words* method, parametrized by four words: ”virus”, ”flu”, ”AH1N1” (which is special kind of flu widespread in Poland in December 2012), ”disease” and it’s stems, with minimum 50% percent of threshold.

Detectors search for defined patterns within the processed web page body and return the results if all searched criteria are fulfilled. The big advantage of the proposed system is the results memory mechanism. It allows to return the result only if it appears on processed web page for the first time, or if the surrounding content has changed since the previous processing.

TABLE I
Crawl performance

Domain	NoC/FC	GS	AFU/APU	ACT	EFF	UM/PM
interia.pl	17/11	3,8mln	3320/3150	1h 45m	0,5	0/0
gazeta.pl	17/12	3,6mln	4375/3951	1h 52m	0,58	24/283
onet.pl	15/10	2,4mln	6112/5950	3h 10m	0,52	119/286
wp.pl	17/8	6,9mln	2602/2520	1h 45m	0,4	258/581

- NoC - number of crawls
- FC - number of full crawl processes
- GS - number of pages returned by Google ”site:” query
- AFU - average number of urls found on domain during single crawl
- APU - average number of urls processed on domain during single crawl
- ACT - average total domain crawl time
- EFF - average crawl efficiency [urls/second]
- UM - number of urls with with any patterns matched
- PM - number of patterns matched within the domain

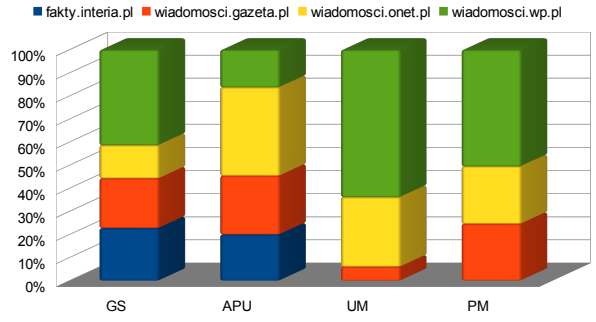

Fig. 3. Crawl performance diagram

The test period last 48 hours – it started on Saturday at 10am and it finished on Monday at 10am. Crawl performance

recorded during the test are shown in Table I and Figure 3. There could be observed some interesting facts:

- Number of urls within the domains returned by the Google "site:" search query varied from 2,4 to almost 7 millions. The number is huge, because it includes archive urls from the Google storage. Performed crawl returned at most about 6000 urls on single crawled domain. This is less than 0.3% of urls declared by the Google search engine, but those urls are all currently available urls, accessible by links spread from top domain url.
- Time of the single domain crawl process varied from 1h30min to 4 hours per domain. Average domain crawl efficiency was comparable for all the domains, about 0,5 processed url per second.
- Not every crawl process succeeded with full url list being processed. Some crawls has been terminated after crawling just a few percent of urls from domain. This is probably a result of temporary ban for IP of the crawling system.
- The number of returned results is neither related to the estimate (returned by Google "site:" query), nor real urls number of the domain. Most results were found on the smallest of crawled domains. The biggest domain was on the second position regarding number of unique web pages with matched pattern.
- One of the domains (fakty.interia.pl) did not contain any results for whole test crawl period.
- About 95% (average) of urls found during crawl process are the documents in textual (text or html) format and they are processed by the detectors.

Crawl and detection effects are presented in Table II. Results are grouped into two hour time units. First two rows of table indicate hour of test ("HoT") and related hour of day ("HoD") of crawl process. "NoU" row represents a number of unique urls found by all detectors within every domain crawled during the test. "NoP" represents a number of patterns found on web pages by all detectors. Below, there are statistics for all detectors separately, displayed as "NoU" and "NoP" values.

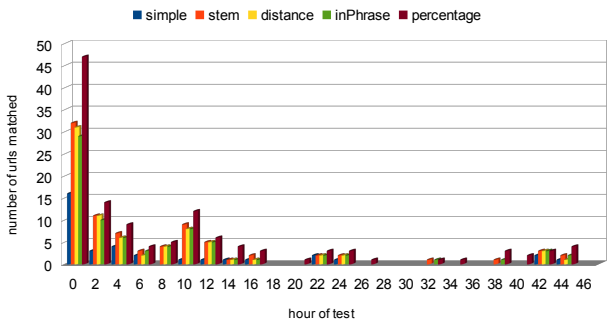

Fig. 4. Number of web pages found during crawl

Interesting remarks that could be observed on a base of test results are:

- The detector with biggest number of urls returned is the *Percentage* one – based on *List of optional words*. It is also result of its search criteria - 2 of of 4 words were required to report as pattern matched.
- Detectors based on word's stems (Stem/Distance/InPhrase) return very similar number of results (about 90% of average common results).
- Detector Simple (without word's stems) returns about 40% of stem-based detector results and matches about 30% of patterns found by stem-based detectors.
- In the figure 4, there is clearly shown, that most of the results (above 51%) are found during first crawl process (in first 4 hours of test).
- There could be observed some tendencies and periodicity of new information appearance. New patterns are found every morning, between 22 and 26 hour of test (8-12am), and also between 42 and 46 hour of test (4-8am). It differs slightly but it is probably caused by crawl density process. Also Saturday afternoon is the time, when the information peak could be observed. Both of this remarks could be result of:
  - new articles published (morning news),
  - increased activities of users commenting articles (evenings).
- an average number of detected pattens observed in the Figure 5 correspond to the trend of the number of unique urls containing pattern. Although there can be observed some deviations. On the 6 and the 16th hour of crawl, there are conspicuous peek of number of detected patterns, which is not related to an adequate incrementation of number of unique pages containing results (8 pages with 79 results and 14 pages with 210 results). Such result was caused by crawling the portal's search engine webpages, containing set of queries and query results related to search topic.

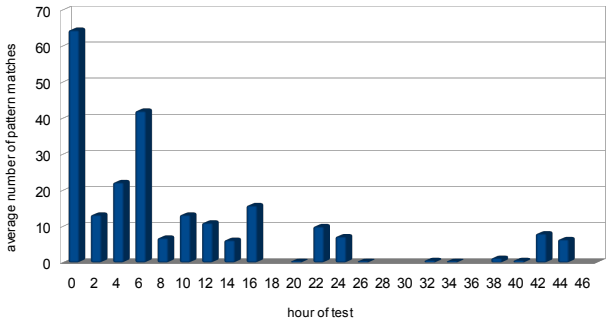

Fig. 5. Number of patterns matched on web pages during crawl

Table III presents statistics about number of patterns found on single web page. The remarks based on those results are:

TABLE II
DETECTOR RESULTS

HoT	0	2	4	6	8	10	12	14	16	18	20	22	24	26	28	30	32	34	36	38	40	42	44	46
HoD	10	12	14	16	18	20	22	00	02	04	06	08	10	12	14	16	18	20	22	00	02	04	06	08
NoU-Simple	16	3	4	2	0	1	1	1	1	0	0	2	1	0	0	0	0	0	0	0	0	2	1	0
NoP-Simple	24	3	11	11	0	4	4	4	3	0	0	7	4	0	0	0	0	0	0	0	0	5	4	0
NoU-Stem	32	11	7	3	4	9	5	1	2	0	0	2	2	0	0	0	1	0	0	1	0	3	2	0
NoP-Stem	67	15	25	59	8	16	13	7	31	0	0	10	8	0	0	0	1	0	0	1	0	10	8	0
NoU-Dist	31	11	6	2	4	8	5	1	1	0	0	2	2	0	0	0	0	0	0	0	0	3	1	0
NoP-Dist	63	15	19	36	8	11	10	4	3	0	0	8	7	0	0	0	0	0	0	0	0	7	4	0
NoU-InPhr	29	10	6	3	4	8	5	1	1	0	0	2	2	0	0	0	1	0	0	1	0	3	2	0
NoP-InPhr	52	13	19	36	8	11	10	4	3	0	0	8	7	0	0	0	1	0	0	1	0	7	5	0
NoU-Perc	47	14	9	4	5	12	6	4	3	0	1	3	3	1	0	0	1	1	0	3	2	3	4	0
NoP-Perc	116	20	37	68	10	24	18	12	39	0	2	17	10	2	0	0	1	2	0	4	3	11	11	0
NoU	155	49	32	14	17	38	22	8	8	0	1	11	10	1	0	0	3	1	0	5	2	14	10	0
NoP	322	66	111	210	34	66	55	31	79	0	2	50	36	2	0	0	3	2	0	6	3	40	32	0

- HoT - hour of a test (0 - 47)
- HoD - hour of a day (0 - 23)
- NoU - number of unique urls with results (for all detectors)
- NoP - number of pattern occurrences (for all detectors)
- NoU-X, NoP-X - NoU or NoP for particular detector

TABLE III
NUMBER OF PATTERNS FOUND ON WEB PAGE

Number of patterns on page	1	2	3	4	5	6	7	8	9	10	11	12	13	14	28	34	35	57	65
Number of occurrences	203	93	28	30	8	6	10	10	2	1	1	1	1	1	1	2	1	1	1

- above 50% contained only one pattern within its body,
- almost 25% contained 2 pattern matches within its body,
- about 15% of web pages contained 3 or 4 pattern matches,
- 9% of results contained 5 to 8 results within its body,
- less than 3% of results contained more than 8 pattern occurrences.

The tests showed clearly, that the solutions used in the presented system works correctly. The performance was satisfactory during whole two-day long experiment. The changes detection algorithm was able to find fragments of Web pages, which have actually changed between visits. The detectors using the stemmer algorithm have demonstrated advantages over a simple, word-based query language.

V. CONCLUSIONS AND FURTHER WORK

The system presented in this paper is a promising base for further works and development, in the area of information retrieval from WEB resources. After certains improvements it could provide useful functionalities, that can definitely find applications in real-life scenarios. Created method for detecting modifications in the webpages' content provides an efficient way of finding only newly-added information.

The proposed solution may also have a variety of applications in open source intelligence analysis. The crawler can be used to build a knowledge base that contains information about objects, events and relations between them (e.g. companies and involved people). This can be further integrated with other analytical tools, such as LINK platform, which supports mainly criminal analysis [13]. This way the crawler can be used as valuable data source for performing various analyses (for example searching relations between people and companies involved in fraud).

Further research on the approach will include performance improvements and development of more advanced methods for defining content patterns. Moreover, tags-avoiding methods are planned, in order to optimize results quality. A possibility of defining semantic meaning of a content or of a similarity to a given text would be more useful than specifying a list of words. This could be achieved by methods of machine learning [14] for building classifiers for particular types of content

ACKNOWLEDGMENT

The research leading to these results has received funding from the research project No. O ROB 0008 01 "Advanced IT techniques supporting data processing in criminal analysis", funded by the Polish National Centre for Research and Development.

REFERENCES

[1] M. Kunder, *WorldWideWebSize.com*, 12.2012
[2] Alexa – provider of global web metrics, *http://www.alexa.com/*, 01.2013.
[3] J. Alpert, N. Hajaj, *We knew the web was big...*, http://googleblog.blogspot.com/2008/07/we-knew-web-was-big.html, 25.07.2008
[4] L. Liu, W. Tang, D. Buttler, C. Pu. *Information Monitoring on the Web: A Scalable Solution* World Wide Web, 2002, Volume 5, Issue 4, pp 263-304
[5] Liu, L., Pu, C., Tang, W.; *WebCQ-detecting and delivering information changes on the web.* In Proceedings of the ninth international conference on Information and knowledge management (pp. 512-519). ACM, 2000
[6] V. Ramasubramanian, R. Peterson, E.G. Sirer, *Corona: A high performance publish-subscribe system for the world wide web.* Proceedings of Networked System Design and Implementation (NSDI). 2006
[7] F. Menczer, R.K. Belew. *Adaptive Information Agents in Distributed Textual Environments*, Proceedings of the 2nd International Conference on Autonomous Agents, ACM Press, 1998, p. 157-164.

[8] H. Dong, F.K. Hussain, E. Chang. *State of the Art in Semantic Focused Crawlers*. Computational Science and Its Applications ICCSA 2009, International Conference, Seoul, Korea, 2009, p. 910–924.

[9] B. Tan, S. Foo, S. C. Hui; *Web information monitoring for competitive intelligence* Cybernetics and Systems, Vol. 33, Iss. 3, 2002

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

[11] W. Turek, A. Opaliński, M. Kisiel-Dorohinicki *Extensible Web Crawler – Towards Multimedia Material Analysis*, Multimedia Communications, Services and Security, 5th International Conference, MCSS 2011, Krakow, Poland, 2011, p. 183–190.

[12] K. Wilaszek, T. Wjcik, A. Opaliński, W. Turek. *Internet Identity Analysis and Similarities Detection*, Multimedia Communications, Services and Security, 5th International Conference, MCSS 2012, Krakow, Poland, 2012, p. 369–379.

[13] R. Debski, M. Kisiel-Dorohinicki, T. Milos, K. Pietak, *LINK: a decision-support system for criminal analysis*, MCSS 2010: Multimedia Communications, Services and Security: IEEE International Conference, Springer, 2010, p.110-115

[14] B. Śnieżyński, *Resource Management in a Multi-agent System by Means of Reinforcement Learning and Supervised Rule Learning*, Springer Lecture Notes in Computer Science vol.4488, 2007, p. 864–871.