DOI:10.2298/CSIS110918036P

WebMonitoring Software System: Finite State Machines for Monitoring the Web

Vesna Pajić[1], Duško Vitas[2], Gordana Pavlović Lažetić[2], and Miloš Pajić[1]

[1] University of Belgrade, Faculty of Agriculture, Nemanjina 6,
11080 Zemun, Belgrade, Republic of Serbia
svesna@agrif.bg.ac.rs, paja@agrif.bg.ac.rs
[2] University of Belgrade, Faculty of Mathematics, Studentski trg 17,
11000 Belgrade, Republic of Serbia
vitas@matf.bg.ac.rs, gordana@matf.bg.ac.rs

Abstract. This paper presents a software system called WebMonitoring. The system is designed for solving certain problems in the process of information search on the web. The first problem is improving entering of queries at search engines and enabling more complex searches than keyword-based ones. The second problem is providing access to web page content that is inaccessible by common search engines due to search engine's crawling limitations or time difference between the moment a web page is set up on the Internet and the moment the crawler finds it. The architecture of the WebMonitoring system relies upon finite state machines and the concept of monitoring the web. We present the system's architecture and usage. Some modules were originally developed for the purpose of the WebMonitoring system, and some rely on UNITEX, linguistically oriented software system. We hereby evaluate the WebMonitoring system and give directions for further development.

Keywords: finite state automata, finite state transducers, software, web monitoring, electronic dictionaries, web search

1. Introduction

The problem of effective search for some particular piece of information is quite current, because of the tremendous amount of information available on the World Wide Web. Natural Language Processing (NLP) and Computational Linguistics (CL) play a dominant role in attempts to solve this problem. Depending on the structure of an electronic text, properties of the language it is written in, and user requirements, NLP and CL have different levels of success. Although very fast and powerful search engines already exist, there are still problems that remain unsolved. In our research, we focused on the two of them.

First, there is a problem of "invisible web" [1], i.e., not having access to some content on the web through search engines. Almost every modern

search system (Google[1], Yahoo[2] etc.) consists of several sub-systems, the most important being the sub-systems for crawling, indexing, and ranking web pages. Since search systems are faced with a huge number of web pages to download and process, each subsystem imposes certain limitations. For the search engine's crawler to find a web page, it is necessary either that the web developer has notified the system of its existence on the web by registering the page URL to the search engine, or that the crawler has visited some other web page containing the hyperlink that points to it. Otherwise, the page remains "invisible" to the search system, and therefore to users. In addition, given that resources of every search system are limited, each crawler has certain limitations that keep crawling process within the limits of available resources. Some search engines limit the total number of pages in the index and drop the old pages when there are new ones, while others limit the frequency of repeated visits to pages. Whenever the search engine decides to limit the search process, the part of the information remains unavailable to users.

There is also a time difference between the moment some content is uploaded to the web and the moment the crawler finds it. This is a major problem for pages that frequently change their content, such as daily news or different forums. Some search engines, including Google, allow the author of a website to reduce this interval, i.e. to increase the frequency at which the crawler visits the site in order to better respond to customer requirements. Another way of overcoming the problem of dynamic content is the concept of RSS ("Really Simple Syndication"), which implies that the author of a particular web site edits and maintains a list of changes on the web site. This list is called "RSS feed". Customers interested in following the changes on the web site can access this list automatically by using a special program known as RSS aggregator. In either case, the visibility of the content depends on the website author. If the author has not provided an RSS feed, nor reduced the interval of the crawler's visits to the page, the users can do no more but personally and regularly check the contents of a particular website in searching for specific information.

The second problem is the way search engines queries are composed, which is often not adequate. Regardless of the facts that Internet users come from different countries, have different areas of interest and levels of education, speak different languages, and have different needs for information, they are all facing the same or similar forms when querying search engines. Mostly, it is a HTML form for keyword search, sometimes with advanced options that allow users to further limit their search. There is no way for a user to formulate some more complex queries.

This problem is even more evident in case of morphologically rich languages, such as Serbian language. For example, if users want to find web pages about Serbian national football team, they would be interested in documents that contain some of the phrases:

1 http://www.google.com

2 http://www.yahoo.com

reprezentacija Srbije („national team of Serbia");
naša reprezentacija („our national team");
reprezentativci Srbije („football players of national team of Serbia");
fudbaleri Srbije („football players of Serbia");
tim „orlova" („eagles" team);
naš nacionalni tim ("our national team");
tim Radomira Antića ("Radomir Antic's team");
srpski fudbaleri ("Serbian football players");
srpska ekipa ("Serbian team");
srpska reprezentacija ("Serbian national team").

Even more, the documents containing any of the inflectional forms of the above phrases are also relevant for the user. General-purpose search engines do not allow making such queries. In recent years, Google has made significant efforts to improve its search process and to bring it to customers, so it returns results where keywords appear in inflected forms as well. This is a good attempt to improve search, but the big problem is that users have no control over this process.

In our research, we focused on solving the above-mentioned problems: improving the way of formulating queries, which will allow for describing more complex context of information, and enabling the access to the content on the Web in the shortest possible time interval. The proposed solution uses finite state automata as search queries, which allows users to describe very complex contexts and phrases they wish to find on web pages. We overcame the second problem by designing a special system that supports the concept of monitoring the web, over which the user has full control. As a result, we have developed the software system called WebMonitoring, which works as a client application on a user's computer. It has its own crawling sub-system that allows a user to set a seed URL and a depth level of crawling, and therefore to monitor one page, one part of a web site or the whole web site. Users describe information they want to find by graphs representing finite state automata or transducers. Those graphs are used as a query for the search. Users can set the way they wish to be alarmed if some information occurs on the monitored web page, as well as the interval of repeated checks.

2. State of the Art in Monitoring the Web

In attempts to improve the process of searching for information on the WWW, different tools for searching and monitoring the web have been developed. Depending on their architecture and functionality, as well as the problems they focus on, they can be divided into two groups.

The tools in the first group focus on monitoring the web. They are all designed primarily for notification if a web page has been changed; there is no possibility to search for some complex queries, except queries based on keywords using Boolean operators. A user can set the frequency of downloads, but it is often limited to some minimum interval (in most cases it is

12 hours interval). Changes occurring on the page in less than 12 hours may not be noticed by the system. Some of the tools use regular expressions, but only to restrict the monitored web page content. Examples of such tools are online systems ChangeDetect[3] and WebSite Watcher[4].

The ChangeDetect system is an online tool for monitoring web pages. For each page a user wants to monitor it is possible to set several parameters: frequency of downloads, regular expression filtering, events causing alerting the user and so on. Still, the minimum monitoring interval is 12 hours. Changes occurring on the page in less than 12 hours may not be noticed by the system. Moreover, the system is designed primarily for notification if a web page has been changed. There is no possibility to search for complex queries, nor the possibility to monitor the entire site. While a user can assign multiple pages to be monitored in the control panel, the monitoring system monitors only the pages with listed URL within each process.

WebSite-Watcher is the software designed to track changes on any number of web pages. With this tool it is possible to monitor all formats of electronic text, including even password-protected pages. Moreover, this tool allows a user to monitor changes of binary files in the sense of changing the size or the date the file was changed. However, even this tool does not support setting up complex queries. WebSite-Watcher uses regular expressions, but only to restrict the monitored web page content. A user is enabled only to search for the occurrence of a keyword or a phrase.

The second group of available tools contains tools that focus on making queries for the search. They are often linguistically oriented, and one of the best known is WebCorp system [2]. WebCorp is a software tool designed by the Department of Research and Development of English Language at the University of Birmingham. It is intended primarily for linguistic research, but it can be used for search as well. A user is allowed to search for a specific word, a phrase, or a pattern. Patterns can be formed by combining the operator *, which replaces any sequence of characters in an expression, square brackets and the OR operator. Although this tool represents a significant improvement since it allows describing complex phrases or patterns, it still does not solve the problem since it relies upon results from the existing search engines. Therefore, there is still much information on the web that remains inaccessible by this tool.

A free online concordance service, GlossaNet [3], is a software tool that solves many of the problems related to information search on the web. It is intended for search into dynamic Web corpora. Users define a corpus by selecting RSS feeds in a pre-selected pool of sources. The GlossaNet crawler regularly visits these sources in order to generate a dynamic corpus. A user can register one or more search queries on his/her dynamic corpus, which are represented with finite state automata and graphs. Those queries will be reapplied to the corpus every time it is updated and new concordances will be recorded for the user. The GlossaNet greatly improves the process of search,

[3] http://www.changedetect.com
[4] http://www.aignes.com

but its main disadvantage is the way it makes corpora. The GlossaNet relies upon RSS feed, and therefore upon a web site's owner and his/her decision about what is to be updated. If the owner of a web site does not provide information about some change in page content, the user will not be able to access that information.

3. WebMonitoring Software System

1.1. Theoretical Background

Finite State Machines in NLP. Finite state machines (automata and transducers) are used in many fields of computational linguistics. Their use is justified from the standpoint of linguistics, as well as from the standpoint of computer science. From the linguistics point of view, finite state machines are adequate for describing relevant local phenomena in language research and for modeling some parts of natural language, such as its phonology, morphology, or syntax. Some examples of adequate representation of different linguistics phenomena by finite state machines are given in [4]. From computer science point of view, the use of finite state machines is motivated by time and space efficiency. Time efficiency is achieved by using deterministic finite state machines. The output of the deterministic machines depends mostly on the size of the input, so they are considered to be optimal ([5] and [6]). Space efficiency is achieved by minimizing deterministic machines [7].

Finite state machines can be very complex and difficult to maintain, which leads to some problems in practice. For example, if someone tries to describe the language syntax by finite state machine, the corresponding graph would be very immense, and finding some particular information, such as noun phrases, would be time consuming and impractical. So, instead of one big graph, we use a collection of sub graphs. This method has a strong theoretical background in the theory of Recursive Transition Networks (RTN). RTN are extension of context free grammars ([8] and [9]). The arcs in RTN are labeled with corresponding grammars, while the states are labeled arbitrarily. There are several computer tools for linguistic research based on FSM and RTN ([10], [11] and [12]). Detailed review of theoretical and practical use of finite state transducers in natural language processing is given in [13], [14], [15], [16], [17] and [18].

The Concept of Monitoring the Web. The concept of web monitoring has emerged from the need for automating certain actions taken by user in order to be notified of changes that occur on a particular web page or site. In practice, there are often situations when a user visits a web site expecting that some event occurred on it, without being interested in the rest of the content of the web site. Examples of such events are announcement of the results of

some competition, electronic message with the specific content or from a specific person, the appearance of news about a particular topic, and so on. In such cases, the user regularly, with the schedule that he or she believes is optimal or possible at a time, visits the web site of interest looking for the event. The software for web monitoring automates this searching process and simulates the actions that the human would take.

1.2. Architecture of the WebMonitoring System

The WebMonitoring software system is developed in the order to overcome problems in search process defined in the Section 1. It is written in the Java programming language and consists of several sub-systems:

- the system for making queries – it is based on the Unitex [11] software system
- the management system
- the crawler
- the system for text post-processing
- the alarming system
- the graphical user interface

The description of the software architecture is given in Figure 1.

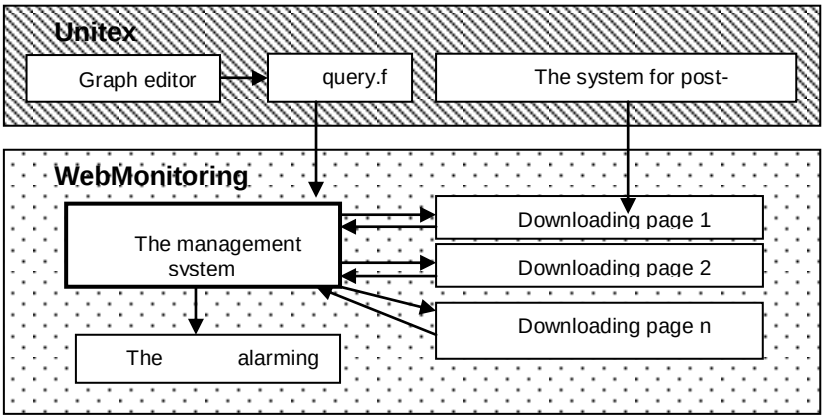

Fig 1. Architecture of the WebMonitoring system

The management system, the crawler, the alarming system and the graphical user interface were developed and written by the authors for the purpose of the WebMonitoring system, while we used the Unitex software and some of its components for querying and post-processing the text.

A user creates a graph that describes an event of interest using the Unitex system. This graph is passed as an input to the management system. Using the graphical user interface of the WebMonitoring software system, the user

sets the URL of the page or website which is being monitored, adjusts parameters, such as dynamics of the visits, the depth of the crawl process (relative to the page from which the crawl will start), the way of alarming, and so on. After that, the management system runs one separate programming thread for each monitoring process set in the management system. The web pages found in the crawling process are saved locally. The system for post-processing analyzes and processes these pages, and tries to find patterns corresponding to the graph. When a pattern is found (and event occurred), the system notifies the user.

The System for Making Queries. The WebMonitoring system uses graphs produced by the Unitex software system [11] as search queries. Unitex is a collection of programs developed for analyzing text written in natural languages, and for applying different linguistic resources and tools to the text. It is an open source software with a very good, functional, and user-friendly graphical interface. Apart from its well-designed graphical user interface for creating graphs, one of the main advantages of the Unitex software is the possibility to use linguistic resources, such as electronic dictionaries and grammars.

Electronic dictionaries contain simple and compound words, together with their lemmas and the set of grammatical codes. They are constructed by teams of linguists for different languages (for English language [19] and [20], for French language [21] and [22], for Serbian language [23] and [24]). Unitex uses electronic dictionaries in DELA format, where each entry is a line of text terminated by a new line, which conforms to the following syntax:

```
apples,apple.N+conc:p
```

The first word (`apples`) is an inflected form of the entry and it is mandatory. In the former example it is followed by the canonical form (lemma) of the entry. This information may be left out if the canonical form is the same as the inflected form. The following sequence of codes (`N+conc`) gives the grammatical and semantic information about the entry. In the former example, code `N` stands for noun, and `conc` indicates that this noun designates a concrete object. The label `p` stands for “plural”.

Although Unitex can process text in different languages, in the first version of the WebMonitoring software it is assumed that Serbian language will be used. For that reason electronic dictionaries for Serbian language are used ([23] and [24]) in the WebMonitoring modules for post-processing the text.

After applying dictionaries and grammars to the text, Unitex creates separate files with simple words, compound words, and unrecognized words. Those files are used in the search process, so one can refer to the dictionary entry from the Unitex by using lexical masks. For example, a user can use the query `<be.V>` that matches all entries having `be` as canonical form and the grammatical code `V`. Thus all occurrences of the verb *to be* (*am*, *is*, *being* etc.) will be recognized by this query. Beside lexical masks, a user can use

morphological filters. For example, the filter `<<ism$>>` matches all words that end with "*ism*" (conservatism, racism, etc.).

By applying lexical resources to the text, as well as combining lexical masks and morphological filters, users can make graphs that correspond to very complex queries. Those graphs in Unitex may have two formats, the format `.grf`, which is intended for the design phase of graphs, and the format `.fst2`, which is compiled version of graphs, intended for further processing and applying to a text.

The graph that corresponds to the phrases about the Serbian national team described in the Section 1 is shown in the Figure 2.

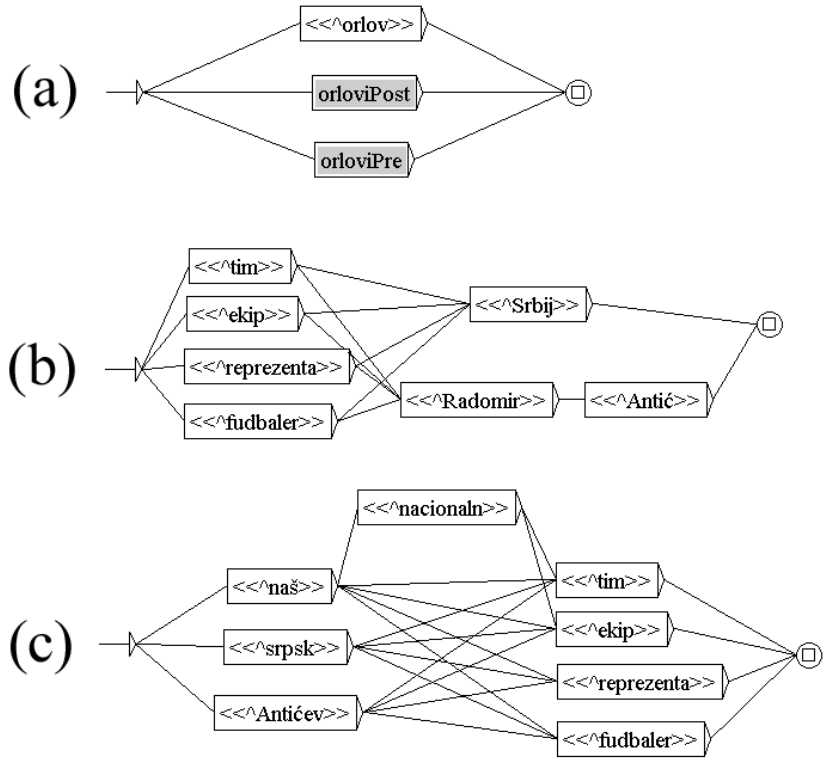

Fig. 2. (a) RTN for describing the phrases corresponding to the Serbian national team; it contains calls to sub-graphs *orloviPost* and *orloviPre*; (b) *orloviPost* is a sub-graph for describing the terms in which the affiliation is given after the noun; (c) *orloviPre* is a sub-graph for describing the terms in which the affiliation is given before the noun

The content of the corresponding `.fst2` file, which is used as a search query, is as follows:

```
0000000003
-1 orlovi
: -2 1 -3 1 1 1
t
f
-2 orloviPre
: 9 2 8 1 6 1
: 5 3 4 3 3 3 2 3
: 7 4 5 3 4 3 3 3 2 3
t
: 5 3 4 3
f
-3 orloviPost
: 5 1 4 1 3 1 2 1
: 11 3 10 2
t
: 12 2
f
%<E>
%<<^orlov>>
%<<^reprezenta>>
%<<^fudbaler>>
%<<^tim>>
%<<^ekip>>
%<<^srpsk>>
%<<^nacionaln>>
%<<^Antićev>>
%<<^naš>>
%<<^Srbij>>
%<<^Radomir>>
%<<^Antić>>
f
```

The `.fst2` format is strictly defined by the Unitex software. The first line represents the number of graphs that are encoded in the file. Lines containing the number and the name of the graph identify the beginning of each sub-graph. In the above file, those are the lines `-1 orlovi`, `-2 orloviPre` and `-3 orloviPost`. The following lines describe the states of the graph. If the state is final, the line starts with `t` character, and with `:` character if not.

For each state, the list of transitions is a sequence of pairs of integers. The first integer indicates the number of the label or sub-graph that corresponds to the transition. Labels are numbered starting from 0. Sub-graphs are represented by negative integers. The second integer represents the number of the result state after the transition. In each graph the states are numbered starting with 0. By convention, state 0 is the initial state.

From the standpoint of the WebMonitoring users, the Unitex system and its interface for creating graphs represent a system for making queries, i.e. for describing an event a user wants to be notified of. Using the Unitex system, a

user creates a graph that describes the event of interest, and then compiles it into the `.fst2` format. This file contains all the necessary information about the event of interest and, as such, it represents the input data for the WebMonitoring system.

The Management System. The management system of the WebMonitoring software consists of several Java classes, and it represents the central point of the overall system. It is designed so as to enable a user to run more than one independent monitoring process.

Every monitoring process is defined by the following attributes:

- *URL* – a web page URL from which the crawl and the search start;
- *graph* – a location of .fst2 file which describes the searched phrases;
- *levels* – an integer that defines the depth of crawl; this attribute is explained in more details later;
- *alarm* – a string attribute that defines the way a user should be alarmed if the event occurred. There are two possibilities: sending an e-mail message and saving the page on the local hard disk. This attribute has the following form: `address; location`, where `address` is an e-mail address for sending the message, and `location` is a directory path for saving the page. If any of these parts of the alarm attribute equals *null*, there will be no alarming of that kind;
- *timeInterval* – an integer that represents a time interval (in milliseconds) between two repeated search processes.

There is an instance of the class *MonitoringProcess* for every monitoring process in the programming code. The class *MonitoringProcess* extends the *Thread* class in Java, i.e. it is a runnable thread. In that way the independent and parallel monitoring of different pages and searching for different queries are allowed.

When the application is started, the main window of the management system will open (Figure 3). It will show the monitoring processes that were started during the previous run in the table, if any. A user can select some process from the table to view or change its characteristics. From this window the user can also delete, make a new, start or stop a process.

The Crawler. After the user has started a monitoring process, the crawler starts a crawl from the URL that is set in the process properties. The crawling process depends on the value of the *levels* parameter. If the value of the parameter is 1, the system should take only this page.

Otherwise, when one page is downloaded from the Internet, it is analyzed to find other hyperlinks on it. The system stores found hyperlinks in the crawl queue and sends them back to the crawler, with the crawling level decreased by one. The crawl process is stopped when all the pages with the level greater than 0 have been processed.

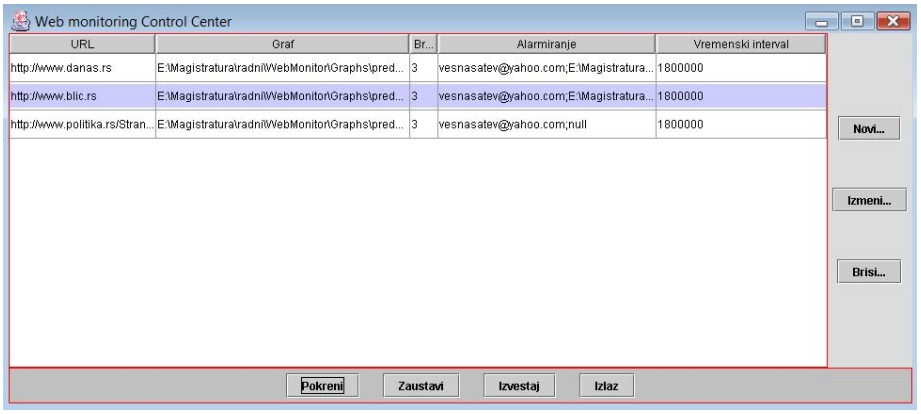

Fig. 3. The control center of the management system

The crawler of the WebMonitoring system is designed to satisfy the following basic principles of a good crawler [25]:

- efficiency - a number of hyperlinks a crawler needs to process increases exponentially with the number of visited pages increasing, so the system must be able to handle the list of addresses in efficient manner from the point of memory usage;
- duplicates – the crawler needs to add to the address list only those addresses that have not been visited, i.e. to recognize the addresses of pages that have already been processed and not to add them to the list;
- politeness – the crawler must comply with the directives contained in the *robots.txt* file on the web server, and to avoid downloading too many pages from a website, so its functionality should not be threatened;
- avoiding the traps - crawler must be able to recognize and avoid sites that are created with the intent to cause an infinite loop for crawlers visiting them.

The crawling sub-system consists of several classes, all belonging to the package *crawler.driller*:

- class *_GO_PARAMS* keeps all the important parameters for the crawling process, such as the seed URL, the number of levels for crawling, the array of the web addresses from the starting page to the current one, etc.;
- class *DrillingQueue* defines a dynamic data structure for keeping the queue of web URL's which have to be downloaded and processed;
- class *PathFinder* keeps information about the path crawler used to get some page;
- class *Driller* starts downloading pages from the seed URL, regarding the number of levels (the *levels* parameter).

The starting class of the crawling sub system is the *CrawlerShell* class, which performs all the necessary adjustments of parameters and runs the crawler. There are many parameters within this class intended for configuring the system for downloading pages. For the purpose of the WebMonitoring

system, some of these parameters have default values that cannot be changed by the user. Thus, for example, the value of the parameter *timeout* is 60 seconds, which means that the crawler sub system will wait for up to one minute for downloading a page from the Internet. Similarly, the maximum time to download pages from one site (in a monitoring process) is defined by the *maxTimeout* parameter, and it is set to 20 minutes (1200 seconds). The declaration section of the CrawlerShell class is shown in the following code:

```
  /** Class for downloading pages starting from the
seed URL */
  public class CrawlerShell{

  // default argument values
    static String hunt = null;
    public static boolean thisSiteOnly = false, silent
= false;
    public static int maxTimeout = 1200, timeout = 60;
    public static int maxlinks = 1000, maxretries = 3;
    public static int levels = 0, sleepParamMS = 500;
    static boolean analyze = true, passedAskMe = false;
    static boolean flash = false, displayLinks = true;
    static magicPath = false;
    static long timeStart = System.currentTimeMillis();
    static boolean logHttp = false, hideUrls = true;
    ......
```

After adjusting the parameters, the crawler for the given URL is started. The text found on every web page in this process is sent to the post-processing system for further analysis, i.e. for the graph search.

The System for Text Post-processing and Alarming. When a web page is downloaded from the Internet, first it is necessary to prepare the text it contains, and then to search for the appropriate event defined by the search graph. This task is performed by the system for post-processing and its class *MonitoredText*. Since the Unitex's external program *Locate* has the central place in the search process, the text from the page is prepared in accordance with the requirements of this program.

The text found on the web page is saved in a text file and stored in a temporary directory. This file is the starting file in the text processing. Although text on web pages is coded differently, most websites in Serbian language nowadays use UTF-8 encoding. UTF-8 (*8 bit Unicode Transformation Format*) encodes each Unicode character as a variable number of 1 to 4 octets, where the number of octets depends on the integer value assigned to the Unicode character. Since each character in the range of U+0000 through U+007F is represented as a single octet, UTF-8 is a very efficient encoding schema of text documents in which most characters are US-ASCII. This is also the reason why this encoding became dominant for electronic mail and web documents, and therefore WebMonitoring system

assumes that the page is encoded in UTF-8. The text found on the page is saved in a text file using methods from the class *fr.umlv.unitex.io.UnicodeIO*. This class is specially designed for the Unitex system purposes, so every file made during the post-processing can be read and processed adequately.

The first step in the post-processing is normalization of the text. Normalization is a process in which the normalization of separators of the text is made, although it is possible to normalize the text on the basis of some other specific rule. Separators in a text are space characters, tabulators, and new lines. When the text is taken from a web page, it is possible that it contains several separators placed side by side in a sequence. These kinds of sequences of separators are being replaced by one space character in the process of normalization. The process of normalization is performed by the Unitex's external program *Normalize*.

The second step of post-processing is the process of text tokenization, i.e. breaking the text up into lexical units. In order to tokenize the text, the Unitex's external program *Tokenize* is called within the class *MonitoredText*.

The electronic dictionaries are applied to a text by the Unitex's external program *Dico*. This step provides the possibility to use morphological and lexical filters in search queries.

After the text is processed in the above described manner, the next step is to search for the patterns described by the user's graph. The search is done by the Unitex's external program *Locate*, which applies a particular automaton or transducer described by a graph to the text and creates an index of found phrases. The program Locate creates two files: `concord.ind` with the references to occurrences found in the text, and `concord.n`, with a number of occurrences and a percentage of recognized tokens. These two files are saved in the working directory, and are used by the sub system for alarming. Method *foundConcordances()* of the class *MonitoredText* uses the file `concord.n` to read the number of occurrences of the searched phrases. If this number is greater than 0, i.e. if the phrases that correspond to the search graph are found in the text, the method *alert()* of the class *MonitoredText* is called, and the user is informed about the event. The way of alarming depends on the values of attributes `email` and `location`. The user can choose to be alerted by an e-mail message, or just to locally save the web page [26].

4. A Use Case

The WebMonitoring software system can be used for different tasks, such as press clipping, detecting spam messages by monitoring electronic mailboxes, management of various documents collections, and so on. We will describe one possible use of the WebMonitoring system.

A user wishes to find all articles related to the current president of the Republic of Serbia, Mr. Boris Tadic, which are or will be published in daily newspapers. The user takes the following steps.

Step 1. Defining the Event to be Searched for. The event to be searched for is an occurrence of phrases regarding Mr. Boris Tadic. Some of the phrases (in Serbian language) are: "*predsednik Tadić*" ("*the president Tadic*"), "*predsednik Srbije*" ("*the president of Serbia*"), "*predsednik B. Tadić*" ("*the president B. Tadic*"), "*predsednik Boris Tadić*" ("*the president Boris Tadic*"), "*Boris Tadić*", as well as their inflected forms ("*predsednika Tadića*", "*Borisu Tadiću*" etc.).

Step 2. Describing the Event by a Graph. The user uses Unitex and creates a graph that describes the defined event. This task can be performed in many different ways, and it depends on user's skills and available resources. One possible way of creating the graph is by using morphological filters. An example is given in the Figure 4. It is necessary to save and compile graph in Unitex, so the file with the extension `.fst2` is created.

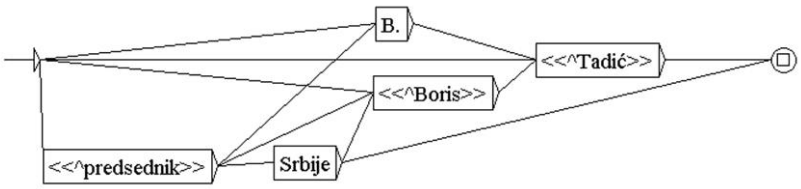

Fig. 4. The query automaton

Step 3. Choosing the Content to Monitor. The user chooses web pages or web sites he/she wishes to monitor. Having in mind that the user wishes to find news articles, he/she chooses official web sites of several daily news papers in Serbia (http://www.danas.rs, http://www.blic.rs and http://www.politika.rs/Stranice/33.lt.html) as starting points of the monitoring process.

Step 4. Creating Monitoring Processes in the WebMonitoring System. In the WebMonitoring system, the user creates a process for each web site he/she wishes to monitor. For each process the user sets URL, number of levels for the crawl, location of the graph describing the event, the way of alarming, and the interval for repeating the process. The main window of the application will appear as shown in the Figure 5.

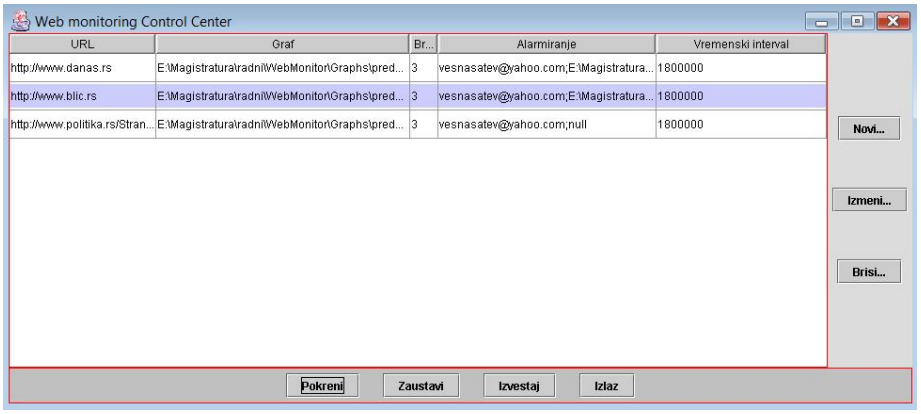

Fig. 5. Control center with three monitoring processes

Step 5. Starting the Processes. The user selects the process in the table showing processes and starts it by choosing the appropriate button. The crawling and monitoring process starts and works in the background. The user can see the progress by choosing the button "*Izvestaj*" ("*Report*"). If the phrase that matches the graph is found on some page, the user is alerted either by e-mail or the page is saved locally on the user's computer.

5. Results and Evaluation of the WebMonitoring System

The case described in Section 4 was used to evaluate the WebMonitoring software system. Since we wanted to evaluate both the possibility to process complex queries and the possibility to access web content in the short time after it appears on the web, we conducted the similar searches with two existing services: GlossaNet, as one of the most powerful, linguistically oriented search system, and Google. We had to change some of the search parameters slightly depending on the requirements and the architecture of these two services, but we tried to keep the search process as uniform as possible.

We monitored two web sites, the official website of the Serbian newspapers *Blic* (http://www.blic.rs) and a popular forum *Krstarica* (Cruiser) in Serbian language (http://forum.krstarica.com). Both web sites have a very dynamic content that frequently changes. Nevertheless, the *Blic* web site provides a RSS feed, while *Krstarica* forum does not.

The results of the evaluation test significantly differ for the two web sites. Comparative characteristics of the three systems are presented in Table 1. The summary of monitoring the *Blic* web site is given in the Table 2, and the summary of monitoring the *Krstarica* forum is given in the Table 3. Since Google retrieves documents (web pages) instead of occurrences of the searched phrases, the number of pages on which the searched phrases are

found is given in the Table 2 and 3. The results are further discussed in the following text, and the results from WebMonitoring system are given in Appendix A.

Table 1. Comparison between GlossaNet, Google and WebMonitoring

System	GlossaNet	Google	WebMonitoring
Type of monitoring	Automatic	Manual	Automatic
Ability to set the monitoring interval	No	No	Yes
Type of query	Finite state graph	Keywords	Finite state graph

Table 2. The results of monitoring the *Blic* web site

System	GlossaNet	Google	WebMonitoring
Number of concordances found by a system	14	850*	51
Most recent result found	Unknown	6 hours	3 minutes

* The number of pages on which the concordances are found; the actual number of concordances is even greater

Table3. The results of monitoring the *Krstarica Forum* web site

System	GlossaNet	Google	WebMonitoring
Number of concordances found by a system	-	43*	14
Most recent result found	-	4 hours	15 minutes

* The number of pages on which the concordances are found; the actual number of concordances is even greater

Search queries. The WebMonitoring software system and the GlossaNet service can process Unitex graphs as search queries, so we used the graph given in the Figure 4. On the other hand, Google can process only keyword searches, so we query Google with: `tadić OR "predsednik tadić" OR "predsednik srbije" OR "boris tadic"`. In that way we were able to control inflectional forms of words with the WebMonitoring and the GlossNet systems, while we did not have any control, nor the possibility to look for different forms of the same word with Google unless we explicitly put it in the search query. In other words, the WebMonitoring and the GlossaNet enable users to search with complex queries, while Google does not.

Monitoring a web site. After defining the search queries, it was necessary to set up web sites to monitor. The content of the chosen web sites was used as a text corpus to search. The GlossaNet system uses a RSS feed from a web site as a mechanism for creating and refreshing a text corpus used in a search process. A user can choose from a predefined list of websites, or create its own corpus, but only for a web site with a RSS feed. After that, the GlossaNet system monitors the chosen site and alerts the user after the monitoring process is over, sending a message with occurrences found. We chose Blic corpus to be monitored during 24 hours. Since Krstarica forum has not got a RSS feed, we could not monitor it with the GlossaNet.

The Google system has its own crawling process in which it downloads and indexes the downloaded pages. Many parameters of the Google's crawling process are automated and recalculated during the process, so a user has no control over the process. Therefore, it is necessary for the user to manually query the Google search system in intervals he/she thinks to be optimal. In our evaluation test, we did two hours check during 24 hours, passing the described search query restricted on the two chosen web sites (adding text "`site:blic.rs`" and "`site:forum.krstarica.com`" to the search query). We also added a 24 hours restriction in order to narrow the search and to achieve a better comparison of results.

The WebMonitoring software system provides the greatest possibilities in terms of setting and controlling the monitoring process. We choose to monitor web sites http://www.blic.rs and http://forum.krstarica.com, with the crawling depth set to 3 during the same 24 hours.

The time interval between two visits to the web site content. The GlossaNet system uses a RSS feed from a web site as a mechanism for refreshing a text corpus used in a search process. The links from RSS feed are visited on a regular basis, but the time period between two visits to links from a RSS feed is defined by the GlossaNet system for corpora building and cannot be changed by a user.

Each time the Google crawling system crawls a URL, it detects whether the resource has changed since the previous crawl. If the resource changed, the change interval is shortened. If the resource did not change, the change interval is lengthened. In that way, a user cannot affect the time interval between two visits. In our evaluation test, we monitored web sites with very dynamic content, and since the Google crawls them with high frequency, the recently added pages from these two web sites were available to Google search. The problem arises when monitoring web pages that do not change their content in a longer time period, and then suddenly change.

The WebMonitoring software system allows a user to set up the time interval, depending on his/her expectations. In our evaluation test, we used 30 minutes as a time interval between two crawls.

The content of a web site "visible" to search systems. Given the way the GlossaNet and the Google download pages from a web site, some contents

remain invisible to them, and therefore to a user. For example, the GlossaNet service downloaded (and searched) only links provided in the RSS feed. Thus, dozens of pages from the Blic web site were not processed. Moreover, Krstarica web site could not be processed at all since it does not provide RSS feed. Therefore, the GlossaNet system was useless in monitoring process of Krstarica forum web site.

In comparison to Google, the WebMonitoring system processed much less pages, but it was expected regarding the architecture of the systems. The Google's crawler and indexing system is far ahead of other crawlers, and we had no intention to compete with Google in it. The advantage of WebMonitoring over Google is in accessing web content in a short time after it appears on the web. In our evaluation test, Google reported one web page as the most recent result, giving the time "26 minutes ago", which would mean that the page was found by Google at 14:50 (Figure 6). On the other hand, the time of publishing given on the page was 8:08 AM 29.02.2012. with changes at 8:55 AM (Figure 7). This means that there is a 6 hours time gap from the time this web page was published on the web to the time Google found it. In our evaluation test, the WebMonitoring system found this page 3 minutes after it was published. In the worst case this time gap can be 30 minutes since the monitoring process is being repeated every 30 minutes. Moreover, a user can additionaly reduce the interval, if needed.

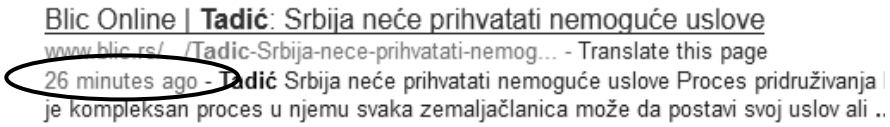

Fig. 6. The most recent Google result: 26 minutes ago (29.2.2012. 14:50)

Fig. 7. The headline as it appeared on the Blic web site, published at 08:08 AM 29.02.2012**.**

Some additional remarks about the WebMonitoring software system are:

- the WebMonitoring crawling system works with high success; there were no errors or situations that some web page could not be downloaded;
- the recognition of phrases corresponding to the graph is complete, i.e. all phrases that existed in the page content and that correspond to the graph have been recognized (we used a sample of 30 pages from *Blic* web site for this testing);
- the speed performance of the system is satisfying having in mind its purpose, although some improvements can be made. This relates primarily to the use of RAM memory instead of saving the changes to the hard disk. Unitex and its external programs record every change of a text on the hard disk, and this practice was continued in the WebMonitoring system, but it significantly slows down the system. Also, the system can be speeded up by using more programming threads. In the current version, one monitoring process is executed within one programming thread, while the inside operations of downloading and processing web pages run sequentially;
- during monitoring of the web site http://www.blic.rs, the event occurred in more than 45% of all web pages. The reason for that is not a significant number of articles about the president of Serbia, but the design of the site. On every web page of the web site there is a section with current news, showing the same news. In the future version of the system this problem should be solved, i.e. the system must be able to recognize the same context on the different pages.

6. Conclusions

This paper considers the improvement of information search process in terms of making more complex queries and access to content of web pages in a short period after their posting on the web. As a solution for complex querying, we suggest using finite state machines. We used finite state machines through the software system Unitex for making queries, but also for the post-processing of the text. We designed and developed the software system called WebMonitoring, which has integrated a subsystem for crawling web pages. With such a system users can do their own crawl and search web pages they wish, independently from the common search engines.

Furthermore, the system WebMonitoring has features that allow user to create, maintain and control processes of monitoring web pages or sites. The system simulates and automates actions a human would take in the process of looking up for some event (a phrase occurrences) on a page.

Since search queries are passed to the system as Unitex graphs, representing finite state automata, this system is not intended for use of a casual user. A basic understanding of finite state automata is required so a user could successfully describe a complex context of searched phrases. Nevertheless, the Unitex' graphical user interface for creating and modifying graphs is user friendly and very intuitive, so any user could easily be trained

to use it. Additionaly, the system could be expanded with modules for automatic transforming regular expressions into Unitex graphs. In that way a user will be able to choose between regular expressions and graphs, depending on his/her skills.

The first version of the WebMonitoring software system should be considered as a demonstration how it is possible to integrate lexical word processing programs with a concept such as web monitoring. Although the current version of the WebMonitoring software system is fully functional and shows positive effects of applying finite state machines to the search process, it is necessary to make some improvements in future versions of the system.

These enhancements are primarily related to elimination of possible errors in the code and upgrading performance and speed of the program. In addition, a user should have more control over the process itself in terms of selecting the language or deciding whether or not to apply dictionaries to the text. The WebMonitoring software system is a general-purpose system. In future versions it is possible to modify the system so as to be specialized for specific types of text (such as medical, technical, etc.), or for special purposes, such as monitoring electronic mailboxes, or search for a specific product in a database accessible from the Internet. We expect that these specialized versions of the WebMonitoring software system will be more efficient.

Nevertheless, the WebMonitoring software system is important because it demonstrates a way of overcoming some problems in the process of information search. It also gives directions for use of linguistic tools in the search process and transcends the limitations in accessing information of the existing search engines.

Acknowledgments. The work presented has been financially supported by the Ministry of Science and Technological Development, Republic of Serbia, Project No. 178006.

References

1. Sherman, C., Price, G.: The Invisible Web: Uncovering Information Sources Search Engine Can't See, Information Today Inc. (2005)
2. A. Kehoe, A. Renouf: WebCorp: Applying the Web to Linguistics and Linguistics to the Web, WWW2002 Conference, Honolulu, Hawaii (2002).
3. C. Fairon, GlossaNet: Parsing a web site as a corpus, Lingvisticae Investigationes, John Benjamins Publishing Company, Volume 22, Number 2, pp. 327-340(14) (2000)
4. M. Gross, D. Perrin, Electronic Dictionaries and Automata in Computational Linguistics, in Proceedings of LITP Spring School on Theoretical Computer Science Saint-Pierre d'Oleron, France, May 25.-29. (1987)
5. D. Vitas, Prevodioci i interpretatori: Uvod u teoriju i metode kompilacije programskih jezika, Matematički fakultet, Belgrade, Republic of Serbia (2006)
6. D. Jurafsky, J. H. Martin, Speech and language processing, Prentice-Hall Inc., 2000.

7. A. V. Aho, J. E. Hopcroft, J. D. Ullman, The Design and Analysis of Computer Algorithms, Addison Wesley, Reading, MA (1974)
8. J. M. Sastre, M. Forcada, Efficient parsing using recursive transition networks with output, In Zygmunt Vetulani, editors, 3rd Language & Technology Conference (LTC'07). 5-7 October 2007. pp. 280–284 (2007)
9. J. M. Sastre, Efficient Parsing Using Filtered-Popping Recursive Transition Networks, Lecture Notes in Computer Science. vol. 5642. pp. 241–244 (2009)
10. B. Olivier, M. Constant, E. Laporte, Outilex, plate-forme logicielle de traitement de textes ecrits. In Proceedings of TALN'06. Leuven, Belgium, UCL Presses universitaires de Louvain (2006)
11. S. Paumier, Unitex 1.2 User Manual, Université de Marne-la-Vallée. http://www-igm.univ-mlv.fr/˜unitex/UnitexManual.pdf (2006)
12. M. D. Silberztein, Dictionnaires électroniques et analyse automatique de textes : le système INTEX. Paris: Masson. (1993)
13. F. Casacuberta, E. Vidal, D. Picó, Inference of finite-state transducers from regular languages, Pattern Recognition, Volume 38, Issue 9, pp.1431-1443 (2005)
14. N. Friburger, D. Maurel, Finite-state transducer cascades to extract named entities in texts, Theoretical Computer Science 313, pp 93 – 104 (2004)
15. J. R. Hobbs, D. Appelt, J. Bear, D. Israel, M. Kameyama, M. Stickel, M. Tyson, FASTUS: A Cascaded Finite-State Transducer for Extracting Information from Natural-Language Text, In Roche E. and Y. Schabes, eds., Finite-State Language Processing, The MIT Press, Cambridge, MA, pages 383-406 (1997)
16. A. Kornai, Extended finite state models of language, Cambridge University Press (1999)
17. E. Roche, Finite state transducers: parsing free and frozen sentences, Extended finite state models of language, Cambridge University Press, pp. 108.-120 (1999)
18. E. Roche, Y. Schabes, Finite-state language Processing, The MIT Press, (1997)
19. A. Chrobot, B. Courtois, M. Hammani-Mc Carthy, M. Gross, K. Zellagui. Dictionnaire electronique DELAC anglais : noms composés. Technical Report 59, LADL, Université Paris 7, (1999)
20. A. Savary. Recensement et description des mots composés - méthodes et applications, Thèse de doctorat. Université de Marne-la-Vallée, (2000)
21. B. Courtois and Max Silberztein, editors. Les dictionnaires électroniques du français.Larousse, Langue française, vol. 87, (1990)
22. J. Labelle, Le traitement automatique des variantes linguistiques en français: l'exemple des concrets, Lingvisticæ Investigationes, 19(1), Amsterdam - Philadelphia: John Benjamins Publishing Company, pp.137–152 (1995)
23. C. Krstev, D. Vitas, Corpus and Lexicon - Mutual Incompletness, in Proceedings of the Corpus Linguistics Conference, Birmingham, (2005)
24. D. Vitas, C. Krstev, I. Obradović, Lj. Popović, G. Pavlović-Lažetić, Processing Serbian Written Texts: An Overview of Resources and Basic Tools, in Workshop on Balkan Language Resources and Tools, 21 Novembar 2003, Thessaloniki, Greece, pp. 97-104 (2003)
25. Baroni, M., Bernardini, S.: BootCaT: Bootstrapping corpora and terms from the Web. In: Proceedings of the 4th International Conference on Language Resources and Evaluation (LREC-2004), Lisbon (2004)
26. V. Pajic, Finite State Transducers in Web Monitoring, Master Thesis, Faculty of Mathematics, University of Belgrade, Republic of Serbia (2010)

Appendix A

The results of monitoring the Blic web site from 9:00 PM 28.02.2012. until 9:00 PM 29.02.2012. (the unique concordances found by the WebMonitoring system):

plomatije Evropske unije Ketrin Ešton i
ozo izjavio je danas posle razgovora sa
Beogradu bio impresioniran posvecenošcu
ki ispit koji položimo”, rekao je
u 241 Nikolic: Ovo nije primena zakona,
e u sudnici i porucio da hoce da pobedi
stervele razgovarace danas u Beogradu s
red optužbom radikala! PUPS bira izmedu
ic: Srbija zaslužuje status kandidata I
Zahtevi Rumunije neopravdani Predsednik
imo na pozitivnoj odluci - rekao je on.
definišu kao rumunska manjina u Srbiji.
iterijumima", dodao je predsednik.
za davanje statusa kandidata Srbiji, a
”, ali ostaje suzdržan optimista.
prava", istakao je predsednik EK.
voren centar NCR korporacije u Beogradu
biti status kandidata za clanstvo u EU.
Briselu predsednik Srbije Boris Tadic.
Rumunije je ozbiljan problem U kabinetu
uaru. U Briselu ce danas i sutra biti i
Prištinu. Ona se zahvalila predsedniku
012. - 22:28h | Komentara: 5 Predsednik
se "smuca" po sudovima? Zašto
Kandidatura nije i ulazak u EU Ešton i
kako je rekla, buducnost svoje zemlje.
i ministri danas doneti odluku o Srbiji
protiv mene vrši Demokratska stranka i
28.2.) biti odobren kandidatski status.
tatusa još par dana? Ketrin Ešton je sa
o na pozitivnom ishodu”, naveo je
kandidatski status. Predsednik Srbije,
eta runda dijaloga Beograda i Prištine.
m Manuelom Barozom. Ocekuje se da ce se
inim zemljama clanicama", rekao je
primamljiv osmeh, Mira Elizabet Fister
apredenje obrazovanja." kaže se u ukazu
ju na tom putu“, izjavio je Tadic. Nade
Ketrin Ešton je sa predsednikom Srbije,
o; Ostale i važne vesti: Povezane teme:
Povezane teme: EU, Žoze Manuel Barozo,
držimo Srbiju na tom putu“, izjavio je
je kosovsko pitanje”, istakao je
ozo na zajednickoj pres konferenciji sa
pozitivnoj odluci, izjavio je u Briselu
io je nemacki ministar. Povezane vesti:
status kandidata uz napredak u dijalogu
protestuju protiv Putina Vesti Politika
asavanje Grcke: Politicari trce u krug »
sije RUBRIKE / Politika / Srbija Srbija
oci pocetka Saveta za opšte poslove EU.

predsednik Srbije Boris Taidc izrazili s
predsednikom Srbije Borisom Tadicem da s
predsednika Srbije Borisa Tadica evropsk
Tadic i dodao da je status kandidata u s
Tadic i DS vrše politicki progon protiv
Tadica i DS. Clan Predsednickog kolegiju
predsednikom Srbije Borisom Tadicem i mi
Tadica i Nikolica! SPO: Mrkonjic podržav
predsednik Srbije Boris Tadic i visoka p
Tadic izjavio je oko 13 casova da je oce
Tadic je istakao da su zahtevi Rumunije
Tadic je izjavio i da ocekuje da ce evro
Tadic je porucio da Srbija ostaje privrž
Tadic je precizirao da je Srbija ispunil
Tadic je rekao da smatra da je Srbija ob
Tadic je rekao da Srbija ocekuje dalje n
Predsednik Srbije Boris Tadic je sa mini
Tadic je, izmedu dva sastanka Saveta min
Tadic je, komentarišuci cinjenicu da se
predsednika Srbije Borisa Tadica jutros
predsednik Srbije Boris Tadic koji bi tr
Tadicu na licnom angažovanju da podrži,
Tadic na predavanju Pitera Bogdanovica P
Boris Tadic nema probleme sa sudovima? N
Tadic Preporuka takode znaci, istakao je
Predsednik Srbije tom prilikom je podset
Predsednik Srbije u Briselu je još jedno
Boris Tadic ", rekao je Nikolic. On
Predsednik Srbije , Boris Tadic, izrazio
predsednikom Srbije, Borisom Tadicem, ra
predsednik Srbije , isticuci da ne &ldquo
Boris Tadic , izrazio je juce u Briselu n
Predsednik Srbije Boris Tadic, koji bora
Tadic , nakon što ministri objave odluku
Tadic , osvrcuci se na zahteve Rumunije.
Tadic , pokušava da opiše sebe. U kratkim
predsednika Srbije Borisa Tadica, povodo
predsednika Srbije, pred današnju odluku
Borisom Tadicem , razgovarala sinoc u Bri
Boris Tadic, Savet ministara EU, Rumunij
Boris Tadic, Srbija, Kandidatura, Odluka
Tadic . Nade predsednika Srbije, pred dan
Tadic . On je ukazao da dobijanje statusa
Tadicem . Samit EU održava se 1. i 2. mar
predsednik Srbije Boris Tadic. Tadic je,
Tadic : Nisam optimista kada je u pitanju
Tadic : Postoji mogucnost da ne dobijemo
Tadic : Srbija ispunila uslove i za datum
Tadic : Srbija neće prihvatati nemoguće
Tadic : Srbija zaslužuje status kandidata
Tadic : Zahtevi Rumunije neopravdani Pred

Vesna Pajić is a teaching assistent at the Department of Agricultural Engineering, Faculty of Agriculture, University of Belgrade, Serbia, since 2003. She received Magister degree in Computer Science in 2010 and currently is doing her Ph.D. dissertation at the Computer Science Department of the Faculty of Mathematics, University of Belgrade. Her research interest includes natural language processing, computational linguistics, text mining, web search and bioinformatics.

Duško Vitas is a professor at the Department of Computing, Faculty of Mathematics, University of Belgrade, Serbia since 1994. Mr. Vitas received his Bachelor degree in Informatics in 1973, Magister degree in 1977, and Ph. D. degree in 1993, all in Mathematics at Faculty of Mathematics, Belgrade. Since 1991 he is employed at the Faculty of Mathematics, Belgrade. He published more than 120 scientific and professional papers.

Gordana Pavlović-Lažetić is a professor at the Computer Science Department of the Faculty of Mathematics, University of Belgrade, since 2009. She obtained her Ph.D. degree in 1988, at the Faculty of Mathematics, University of Belgrade. She spent two years at the University of California, Berkeley, doing research in database and text processing fields. Her current research interest includes databases, data mining, text processing and bioinformatics. She is an author of over 50 scientific papers and participated at more than 30 conferences. Professor Pavlovic-Lazetic supervised a number of Ph.D. and M.S. thesis. She participated in many national and international researches and was a member of organizing and program committees for several conferences. She is a member of ACM.

Miloš Pajić is a teaching assistent at the Department of Agricultural Engineering, Faculty of Agriculture, University of Belgrade, Serbia, since 2002. He received Ph.D. degree in Biotechnical Science in 2012 at the Faculty of Agriculture, University of Belgrade. His research interest includes technical inovations in biosistems, agricultural mechanization and bioinformatics.

Received: September 18, 2011; Accepted: July 18, 2012.