IOSR Journal of Engineering (IOSRJEN) **www.iosrjen.org**
ISSN (e): 2250-3021, ISSN (p): 2278-8719
PP 54-58

Integrated Architecture for Keyword-based Text-Data Collection and Analysis

Nateshia G ,Mrs. Deepa K
Research scholar Providence for women affiliated to Bharathiar University
(Assistant professor providence college for women)

Abstract:*In this paper, we present an integrated framework for keyword-based text data collection and analysis. The integrated framework consists of four types of component: (1) user interface,*
(2) web crawler, (3) data analyzer, and (4) database (DB). The user interface is used to set input keyword and option values for web crawling and text data analysis through a graphical user interface (GUI). In addition, it provides analysis results through data visualization. The web crawler collects the text data of articles posted on the web based on input keywords. The data analyzer classifies the articles into "relevant articles" and "no relevant articles" using predefined knowledge (i.e., a set of words to be included in the articles). Then, it analyzes the text data of relevant articles and visualizes the results of the data analysis. Finally, the DB stores the collected text data, the predefined knowledge defined by the user, and the results of the data analysis and data visualization. We verify the feasibility of the integrated framework through proof of concept (PoC) prototyping. The experimental results show that the implemented prototype collects and analyzes the text data of articles reliably.
***Keywords**: data analysis, integrated framework, intelligent service, text data collection, web crawling.*

I. Introduction

Recently, intelligent services, such as news curtain and preference analysis and recommendation, have received considerable attention from both academic and industry.(1–3) These services commonly use the text data of articles posted on the web to collect information needed for individuals. The text data can be recorded, collected, and analyzed through a "sensor web", which is a special type of web-centric information infrastructure.(4) Especially, data collection and analysis are the most important functionalities of a sensor web. Therefore, web crawling to collect text data and big data analysis to analyze collected text data are widely considered as key enablers of sensor web for such intelligent services.

To date, a number of existing research studies have tried to implement these functionalities using open source programming languages, such as R, Python, and Scala.(5–7) However, most of them have not used an integrated application for both web crawling and big data analysis. In other words, the functionalities of web crawling and big data analysis have been implemented separately in most of the existing research studies. Therefore, unpredictable delay may occur in the targeted intelligent service, because their feasibility highly depends on the developer's proficiency.(8,9) To enable seamless intelligent services, the design of an integrated architecture encompassing various functionalities (e.g., web crawling, data analysis, and user application) is required.(10)

In this paper, we propose an integrated framework of web crawling and data analysis for keyword-based text data collection and analysis to enable seamless intelligent services. The proposed framework includes the following four components: (1) user interface, (2) web crawler, (3) data analyzer, and (4) database (DB). These components interact with each other to exchange data. The user interface component helps users set the input keyword and detailed option values for web crawling and text data analysis through the graphical user interface (GUI). Moreover, it provides various data visualization results, such as word clouds and word frequency graphs, based on the results of the text data analysis. The web crawler component collects the text data of articles posted on the web and provides the datasets to be used for analysis by storing the collected text data in the DB. The data analyzer component performs the data preprocessing and the data analysis using the datasets provided from the web crawler component. For the data preprocessing, article classification is conducted. Specifically, the articles are classified into "relevant articles" and "no relevant articles" using predefined knowledge (i.e., a set of words to be included in the article). Relevant articles are collected articles that are highly associated with the topic the user intends to search, while no relevant articles are not highly associated with the topic. The data analysis is performed in two steps. The former is a word extraction step in which words composed of three or more characters are extracted based on the text data of relevant articles. The latter is a word filtering step that removes unnecessary words from the extracted words. The results of the data analysis are visualized in the form of word clouds and word frequency graphs. Finally, the DB component

consists of three DBs: crawling DB, predefined knowledge DB, and analysis DB. Each DB stores the collected text data, predefined knowledge, and results of data analysis and data visualization.
We verify the feasibility of the integrated framework through proof of concept (PoC) prototyping. The web crawler and data analysis are implemented using the functional packages provided by open source R, and the interface is implemented using the Java Swing framework.(11,12)
The experimental results show that the integrated framework reliably provides web crawling and text data analysis functionalities.The rest of this paper is organized as follows. In Sect. 2, the detailed design of the integrated framework is described. The implementation and experimental results are presented in Sect. 3. Finally, we conclude the paper in Sect. 4.

II. Functional Architecture of Integrated Framework

Figure 1 shows the functional architecture of the proposed integrated framework consisting of the user interface, web crawler, data analyzer, and DB components. The user interface component consists of three function blocks: user input panel, result view panel, and predefined knowledge panel. The user input panel is used to set the input keyword and option values, including the range of crawling pages, word cloud minimum frequency, and word frequency ranking. The range of crawling pages is the range of web pages on which to crawl the text data of articles. The word cloud minimum frequency refers to the minimum frequency of words displayed in the word cloud, which is one of the data visualization results. Note that the word cloud is an image composed of various words with different sizes depending on their frequency. The frequent word ranking is a criterion of word frequency to determine the words displayed in the word frequency graph. The result view panel is used to provide data visualization results after analyzing the data, and the predefined knowledge panel is used to add or delete the words stored in the predefined knowledge DB of the DB component.

The web crawler component consists of the data collector and file creator function blocks. The data collector performs parsing, language support, and extraction. Parsing involves searching the web pages of articles and crawling the text data of articles on specific web pages. For this, it creates a uniform resource locator (URL) to search the web pages of articles based on the input keyword defined by the user and counts the number of web pages of articles using the corresponding URL. In addition, during parsing, it creates a URL for the web pages to crawl the articles based on the number of searched web pages of articles. Then, it requests a hypertext markup language (HTML) file of all the articles contained in the webpage using the corresponding URL. Language support involves encoding in the UTF-8 type to prevent data loss during web crawling. Extraction involves extracting the text data of the articles from the HTML file. The file creator creates a data file that records the extracted text data and stores the file in the crawling DB of the DB component.

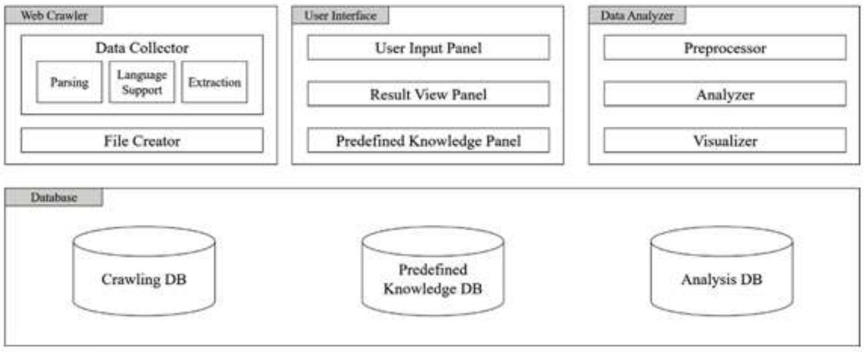


Fig. 1. Functional architecture of integrated framework.

The data analyzer component consists of three function blocks: preprocessor, analyzer, and visualizer. The preprocessor preprocesses the extracted text data of the articles stored in the crawling DB. The preprocessor classifies the articles into relevant and no relevant articles by comparing the text data of the articles with a set of words in predefined knowledge. If the article contains one of the words in predefined knowledge, the preprocessor function block classifies it as a relevant article. The text data of relevant articles is stored in the analysis DB of the DB component. On the other hand, the text data of no relevant articles is removed. The analyzer performs two steps: word extraction and word filtering. In the word extraction step, words composed of three or more characters are extracted based on the text data of relevant articles. In the word filtering step, unnecessary words (e.g., “a”, “an”, “the”, and “that”) are removed from the text data of relevant articles. The data analysis results are stored in the analyzer DB of the DB component. The visualizer performs data visualization based on the data analysis results and the option values set through the user input panel. The data visualization results are visualized in the form of word clouds and word frequency graphs and stored in the analyzer DB.

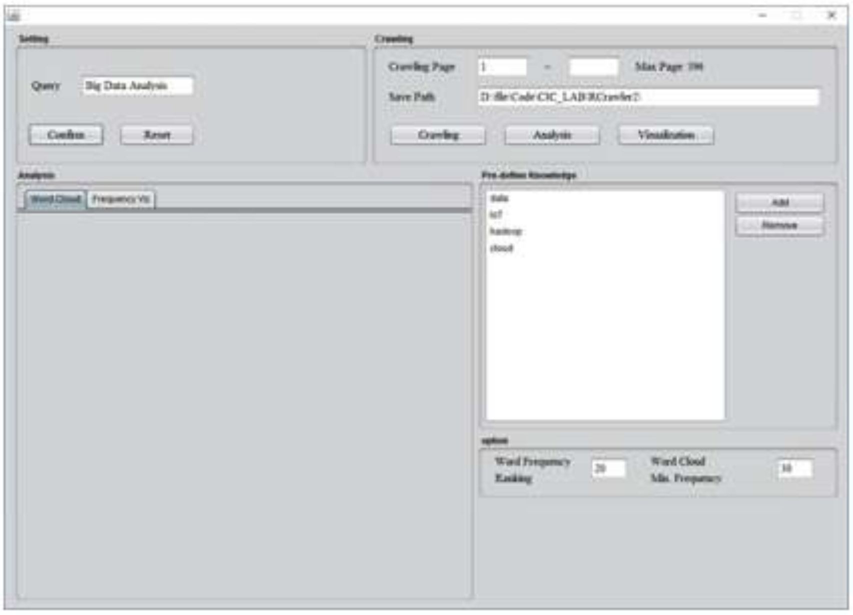
The DB components consist of crawling DB, predefined knowledge DB, and analysis DB. The crawling DB stores the text data of articles collected from the web through the web crawler component. The predefined knowledge DB stores a set of words added by the user through the predefined knowledge panel. The analysis DB stores the text data of relevant articles, data analysis results, and data visualization results.

III. Implementation

The feasibility of the integrated framework is verified through PoC prototyping. For this, the GUI is implemented using the Swing framework provided by the Java development kit (JDK) version 1.8, and the functionalities of data collection and data analysis are implemented using open source R version 3.4.0.

Figure 2 shows the GUI of the integrated framework. The input keyword settings and the request for searching the web pages of articles are executed through the setting group box in the GUI. In the setting group box, the "Query" field is used to set the input keyword, and the "Confirm" button is used to search the web pages of articles based on the input keyword.

The request for searching the web pages of articles is performed as follows. First, the HTML file containing information with respect to the total number of web pages of articles is requested. Then, the text data whose attribute value is <select class = "search-header"> is extracted from the HTML file through the xpathSApply() function to obtain the total number of web pages of articles. Next, blanks and quotes are removed from the extracted text data, and the text data (i.e., the maximum number of web pages of articles) is displayed in the "Max Page" label in the crawling group box. To collect the text data of articles, the range of crawling pages is set through the "Crawling Page" field. Then, the request for web crawling is executed through the "Crawling" button. In web crawling, the HTML file containing the text data of articles on a certain webpage is requested, and the text data whose attribute value is <div class = "storyBody" p> is extracted from the HTML file through the xpathSApply() function. Web crawling is 443
443



Query: Big Data Analysis

Crawling Page: 1 ~

Max Page 398

Save Path: D:\file\Code\CIC_LAB\RCrawler2\

data
IoT
hadoop
cloud

Word Frequency Ranking: 20

Word Cloud Min. Frequency: 10

Fig. 2. (Color online) GUI of integrated framework.

repeatedly executed to include the entire range of crawling pages set by the user. Finally, the text data of all the extracted articles is accumulated and stored in the crawling DB.

The data preprocessing and data analysis are executed through the "Analysis" button. For data preprocessing, the predefined knowledge is used for article classification. The words in predefined knowledge can be added or deleted by the user. The "Add" and "Remove" buttons in the predefined knowledge group box are used to add/delete words to/from predefined knowledge, respectively. The str_detect() function is used to check whether the words in predefined knowledge are included in the text data of the collected articles. If the article contains one of the words in predefined knowledge, it is classified as a relevant article; otherwise, it is classified as a no relevant article. The text data of relevant articles is stored in the analysis DB, while the text data of no relevant articles is removed. Upon completing data processing, data analysis is executed, in which word extraction and filtering are performed. For word extraction, words composed of three or more characters are extracted from the text data of relevant articles using the sapply() function. For word filtering, unnecessary words are removed from the text data of relevant articles using the str_detect() function. The results of data analysis are stored in the analysis DB.

Data visualization is executed by the "Visualization" button. The results of data visualization are displayed in the form of word clouds and word frequency graphs. For this, the option values (i.e., word frequency ranking and word cloud minimum frequency) in the option group box are used. Word clouds are created using the wordcloud() function, and word frequency graphs are created using the barplot() function. The results of data visualization are stored in the analysis DB. For this, we set the keyword to big data analysis and the crawling page to 1–300.

Since each webpage contains ten articles, a total of 3000 articles are collected. The words in predefined knowledge and the filtering words are listed in Table 1. We set the word cloud minimum frequency to 150; thus, the words with frequencies over 150 are displayed in the word cloud. We also set the word frequency ranking to ten to display ten words in the word frequency graph. The detailed experimental parameters are listed in Table 1.

Figure 3 shows a word cloud result. In the figure, words extracted from the articles with frequencies over 150 are displayed. The words with the highest frequencies have the largest font size and are located at the center of the word cloud. The words are represented in different colors according to frequency, but they have the same color when their frequencies are similar. In our experiment, the frequency of "cloud" is the highest among all the words.

Figure 4 shows the word frequency graph indicating the ten words with the highest frequencies among all the words extracted from the articles. The x- and y-axes of the graph represent the word types and the frequency of words, respectively. The labels on the bar graph show the frequency and frequency rate of each word. In the figure, the frequency of "cloud" is 2132, and the frequency rate of "cloud" is 2%.

Table 1 Experimental parameters.

Parameter	Values
Keyword	Big data analysis
Crawling page	300
Predefined knowledge	Data, IoT, Hadoop, Cloud
Filtering words	good, will, then, the, are, with, and, that, this, but, have, has, can, for, you, from, been, more, they, said, what, its, about, how, was, which, their, into, these, when, there, those
Word cloud minimum frequency	150
Word frequency ranking	10

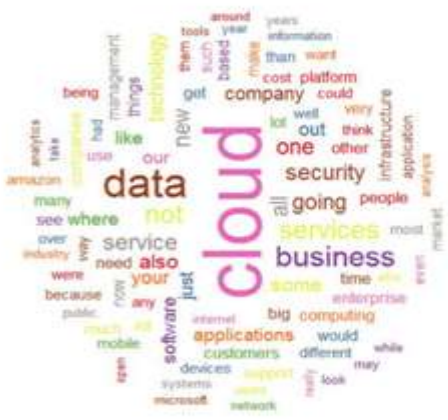


Fig. 3. (Color online) Word cloud.

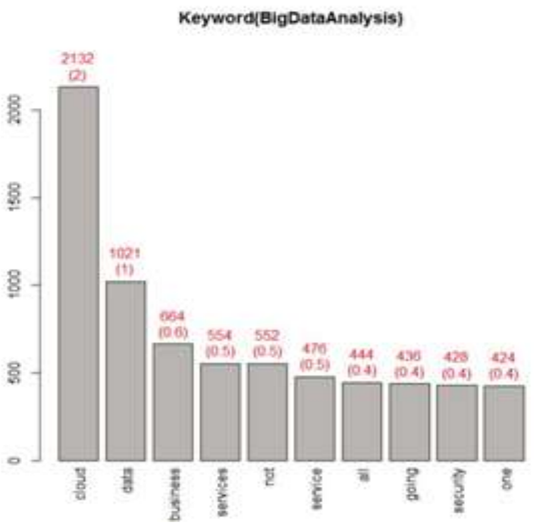


Fig. 4. (Color online) Word frequency graph.

IV. Conclusion

In this paper, we proposed an integrated framework of web crawling and big data analysis for keyword-based text data collection and analysis. The integrated framework consists of four components: user interface, web crawler, data analyzer, and DB. The user interface component helps the user set input keyword and option values through the GUI, and the web crawler component collects the text data of articles posted on web pages based on the input keyword. The data analyzer component performs data preprocessing, data analysis, and data visualization. Finally, the DB component stores the collected text data, predefined knowledge, data analysis results, and data visualization results. To verify the feasibility of the integrated framework, PoC prototyping was conducted. For the experiment, we collected text data of articles from ZDNet based on the user-defined input keyword and analyzed the collected text data to check their frequencies. The data analysis results were visualized in the form of a word cloud and word frequency graph. The experimental results showed that the integrated framework reliably provides the functionalities of web crawling and text data analysis.

References

[1]. C. Dobre and F. Xhafa: Future Gener. Comp. Syst. **37** (2014) 267.
[2]. W. Raghupathi and V. Raghupathi: Health Inf. Sci. Syst. **2** (2014) 1.
[3]. Z. Khan, A. Anjum, and S. L. Kiani: Proc. 2013 IEEE/ACM 6th Int. Conf. Utility and Cloud Computing (IEEE, 2013) 381.
[4]. A. Sheth, C. Henson, and S. S. Sahoo: IEEE Internet Comput. **12** (2008) 78.
[5]. S. Landset, T. M. Khoshgoftaar, A. N. Richter, and T. Hasanin: J. Big Data **2** (2015) 24.
[6]. F. Morandat, B. Hill, L. Osvald, and J. Vitek: Proc. European Conf. Object-Oriented Programming (Springer, 2012) 104.
[7]. B. C. D. S. Oliveira and J. Gibbons: J. Funct. Program. **20** (2010) 303.
[8]. A. Mesbah, A. V. Deursen, and S. Lenselink: ACM Trans. Web **6**(2012) 1.
[9]. Y. Zhang: IEEE Trans. Serv. Comput. **9** (2016) 786.
[10]. S. Wang, C.Zhang, and D. Li: Proc. Int.Conf. Industrial IoT Technologies and Applications (Springer, 2016)12.
[11]. R Foundation: https://www.r-project.org/ (accessed July 2017).
[12]. Oracle: http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html (accessed July 2017).
[13]. ZDNet: http://www.zdnet.com/ (accessed July 2017).