ISSN 1310-8271

JOURNAL

OF THE TECHNICAL UNIVERSITY - SOFIA PLOVDIV BRANCH, BULGARIA

FUNDAMENTAL SCIENCES AND APPLICATIONS

Volume 24, 2018

© Journal of the Technical University - Sofia
Plovdiv branch, Bulgaria
"Fundamental Sciences and Applications"
Vol. 24, 2018

COMPARISON OF PYTHON LIBRARIES USED FOR WEB DATA EXTRACTION

ERDİNÇ UZUN, TARIK YERLİKAYA, OĞUZ KIRAT

Abstract: *There are several libraries for extracting useful data from web pages in Python. In this study, we compare three different well-known extraction libraries including BeautifulSoup, lxml and regex. The experimental results indicate that regex achieves the best results with an average of 0.071 ms. However, it is difficult to generate correct extraction rules for regex when the number of inner elements is not known. In experiments, only %43.5 of the extraction rules are suitable for this task. In this case, BeautifulSoup and lxml, which are the DOM-based libraries, are used for extraction process. In experiments, lxml library yields the best results with an average of 9.074 ms.*

Key words: *HTML, DOM, Web Data Extraction, Python*

1. Introduction

Over the years, due to increased content on the Internet, it has become harder and harder to differentiate between meaningful and unnecessary contents on the web pages. Web data extraction [1] (also known as web content extraction, web scraping, web harvesting, etc.) is the process of extracting user required information from web pages. A large amount of data is continuously created and shared online. Web data extraction systems allow to easily collect these data with limited expert user effort [2]. In this study, we will explain how to access this data from web pages. Moreover, time results of libraries in Python that can be used extraction process are compared.

Web data extraction methods can be classified into three different categories: Wrapper based methods [3], DOM (Document Object Model) based methods [4] and machine learning based methods [5]. Wrapper in this task is a program that extracts data of a particular information from web pages. DOM-based methods utilize structure, tags and attributes of HTML. Machine learning-based methods are on state-of-the-art machine learning algorithm and these methods require labeled data obtained from the DOM. This paper focuses on libraries that can be used for this task. Overall performance in terms of accuracy, time efficiency, memory and processor efficiency varies dramatically depending on the library and algorithms used, even while using libraries classified in the same category.

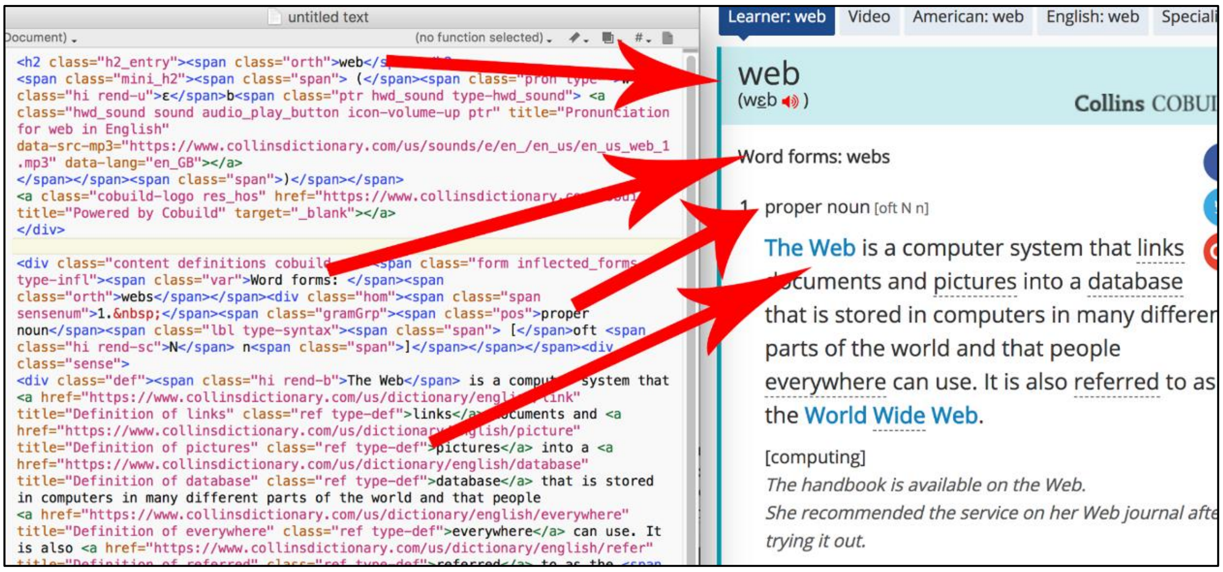
In this paper, we will discuss time efficiencies of web extraction methods including regular expressions and two different data extraction libraries in Python programming language called BeautifulSoup [6] and lxml [7], both can be categorized as a DOM based library.

2. Extraction from a Web Page

DOM [8] is an interface that categorizes each element in an HTML or any other XML-based document into nodes of a tree structure. Each node represents a part of the document and can contain other nodes. DOM standard is handled by World Wide Web Consortium, like HTML. The contents of DOM Tree can be changed by programming languages such as JavaScript, C#, Java, Python and etc.

A web page is downloaded after being requested by a web user via a browser. After downloading, the downloaded document is processed and the DOM elements are produced. Each DOM element is interpreted by the web browser to construct the DOM tree while displaying a web page. DOM elements can also be accessed and modified during the display in a web browser of a web page using client-side JavaScript.

There is a relationship between the DOM and the web page as shown in Fig. 1. This web is from www.collinsdictionary.com that presents dictionaries for English or bilingual word reference and plus thesauruses for expanding your word power. In this web domain, <h2> is an HTML element that defines the most important heading. Moreover, this element have an attribute (class) and value of attribute (h2_entry) that provide additional information for the element.



```
<h2 class="h2_entry"><span class="orth">web</s
<span class="mini_h2"><span class="span"> (</span><span class="pron type
class="hi rend-u">ɛ</span>b<span class="ptr hwd_sound type-hwd_sound"> <a
class="hwd_sound sound audio_play_button icon-volume-up ptr" title="Pronunciation
for web in English"
data-src-mp3="https://www.collinsdictionary.com/us/sounds/e/en_/en_us/en_us_web_1
.mp3" data-lang="en_GB"></a>
</span></span><span class="span">)</span></span>
<a class="cobuild-logo res_hos" href="https://www.collinsdictionary.
title="Powered by Cobuild" target="_blank"></a>
</div>

<div class="content definitions cobuild     span class="form inflected_forms
type-infl"><span class="var">Word forms: </span><span
class="orth">webs</span></span><div class="hom"><span class="span
sensenum">1.&nbsp;</span><span class="gramGrp"><span class="pos">proper
noun</span><span class="lbl type-syntax"><span class="span"> [</span>oft <span
class="hi rend-sc">N</span> n<span class="span">]</span></span></span><div
class="sense">
<div class="def"><span class="hi rend-b">The Web</span> is a comput   system that
<a href="https://www.collinsdictionary.com/us/dictionary/engl    ink"
title="Definition of links" class="ref type-def">links</a    cuments and <a
href="https://www.collinsdictionar    glish/picture"
title="Definition of pictures" class="ref type-def" pictures</a> into a <a
href="https://www.collinsdictionary.com/us/dictionary/english/database"
title="Definition of database" class="ref type-def">database</a> that is stored
in computers in many different parts of the world and that people
<a href="https://www.collinsdictionary.com/us/dictionary/english/everywhere"
title="Definition of everywhere" class="ref type-def">everywhere</a> can use. It
is also <a href="https://www.collinsdictionary.com/us/dictionary/english/refer"
```

web
(wɛb)

Word forms: webs

1. proper noun [oft N n]

The Web is a computer system that links documents and pictures into a database that is stored in computers in many differen parts of the world and that people everywhere can use. It is also referred to as the World Wide Web.

[computing]

The handbook is available on the Web.

She recommended the service on her Web journal afte trying it out.

Fig. 1. *Relationship between the DOM and the web page*

There are a lot of HTML elements such as <span>, <div>, <a>, and etc. in a web page. Fig. 1 shows only a crucial part of a web page that can be used in extraction process. Additionally, these elements have attributes such as id, class, title, href, data-scr-mp3 and etc. id and class are widely used attributes useful for applying styles and manipulating an element with DOM and JavaScript. These information can be used for extraction process. For example, the following rules can be used in this task.

- <h2 class="h2_entry"> : Title of a web page
- <div class="content definitions cobuild br"> : All definitions
- <span class="form inflected_forms type-infl"> : Word forms
- <div class="hom"> : sub definitions

Web pages in a web domain have similar elements. When the appropriate element is selected from a web page, it can be used to extract other web pages of web domain. For example, these rules can be utilized in the other web pages of this domain. In experiments, we have downloaded 30 different web pages for 10 different web domain. Moreover, we have prepared 1-4 rules for every web domain.

3. Regular Expressions and Data Extraction Libraries for Python

Using regular expressions is the well-known technique that can be used in the extraction process. However, it can cause problems when the number of inner tags is ambiguous. In this situation, the DOM-based libraries can be utilized as a solution for these problems.

3.1. Regular Expressions

Regular expressions (sometimes called as regex or regexp) is a sequence of characters to define a search pattern. Regular expressions can be handled using the "re" module in Python. For this module, we firstly prepared extraction pattern shown in Code 1. For example, this code returns the pattern <h2. class..h2_entry.>(.*?)</h1> for an extraction rule of <h2 class="h2_entry">.

Code 1. *Preparing a pattern for a given rule*

```
def prepare_regex(pattern):
    return pattern.replace(" ", ".").replace("\"", ".").replace("'", ".").replace("=", ".") + "(.*?)" + "</" + parse_TagName(pattern) + ">"
def parse_TagName(element):
    return element[(element.find('<') + 1):(element.find(' '))]
```

For an extraction rule, there are two different extraction techniques: the whole document can be searched or the extraction process can be finalized by finding the first record. For some extraction rules, terminating the extraction process after the first extraction can improve the extraction processing time. "re" module supports two different extraction techniques. Code 2 finalizes the extraction process after finding the relevant content.

re.seach method scans through string looking for the first location and return a corresponding match object. If no position in the string matches the pattern, Code 2 returns empty string. However, if more than one record exists, the entire document should be looked at. Code 3 extracts all content from a web page.

Code 2. *Extracting the first record with re*

```
import re
```

```
def extract(html, pattern):
    res = re.search(prepare_regex(pattern), html,
    re.DOTALL)
      if res:
        return res.group(1)
      else:
        return ' '
```

Code 3. Extracting all records with re

```
def extract_all(html, pattern):
    return re.findall(prepare_regex(pattern), html,
    re.DOTALL)
```

re.findall method returns all non-overlapping matches of pattern in a web page, as a list of strings. In experiments, the effect of these extraction techniques will be investigated.

3.2. Data Extraction Libraries in Python

Two well-known extraction libraries, BeautifulSoup and lxml, are used for this task.

3.2.1. BeautifulSoup

BeautifulSoup is a Python data extraction library developed by Leonard Richardson and other open source developers. It is licensed under the Simplified BSD License and works on both Python 2.7+ and Python 3. It can parse HTML and XML documents and provides simple methods to interact with the DOM model.

Code 4. BeatifulSoup extraction methods

```
from bs4 import BeautifulSoup
def extract(html, pattern, parser):
    soup = BeautifulSoup(html, parser)
    return soup.find(parse_TagName(pattern),
    attrs=parse_Attributes(pattern,
    parser)).decode_contents(formatter="html")
def extract_all(html, pattern, parser):
soup = BeautifulSoup(html, parser)
    temp_res =
    soup.find_all(parse_TagName(pattern),
    attrs=parse_Attributes(pattern, parser))
    html_res = []
    for res in temp_res:
        html_res.append(res.decode_contents(formatt
        er="html"))
    return html_res
def parse_TagName(element):
    return element[(element.find('<') +
    1):(element.find(' '))]
def parse_Attributes(element, parser):
    soup = BeautifulSoup(element, parser)
    e = soup.find(parse_TagName(element))
    return e.attrs;
```

BeautifulSoup supports two different extraction techniques like “re” module. Code 4 extracts only the first record in web page.

A BeautifulSoup object has two arguments. The first argument is the source of web page, and the second argument is the parser. The different parsers including html.parser, lxml, and html5lib can be adapted to the object of BeautifulSoup. html.parser comes with Python’s standard installation and provides a class named HTMLParser which can be used as a basic HTML and XHTML parser. lxml is feature-rich and easy-to-use library for processing XML and HTML. It provides Python bindings for the C libraries libxml2 and libxslt and mostly compatible with ElementTree API. It is also open source and licensed under the BSD license. html5lib conforms WHATWF HTML specification which allows the module to parse HTML content the same way a web browser does. It is mainly developed by James Graham and open source under the MIT license.

The find method uses when you only want to find one result. The find_all method scans the entire document looking for results. If the number of extraction result is one content, you can use the find() method for improving time efficiency of the extraction process. In Code 4, the parse_TagName method finds the element name of the pattern and parse_Attributes method returns all attributes and their values in list format. The decode_contents method renders the contents of this element in html format.

3.2.2. Lxml

Some web sites introduced BeautifulSoup [9] recommend to install and use lxml for speed. But also, it can be used stand-alone. lxml supports XPath [10] for extracting the content of a tree. XPath allows you to extract the content chunks into a list. XPath uses "path like" syntax to identify and navigate nodes in an html and xml document. Code 5 returns the XPath expression for a given extraction rule.

Code 5. Preparing of XPath expression for a given element

```
def prepare_XPath(pattern):
    root = etree.fromstring(pattern + '</' +
    parse_TagName(pattern) + '>')
    temp=''
    for att in root.keys():
        temp += '[@'+ att + "='" + root.get(att) +
        "']"
    return ".//" + root.tag + temp
```

For example, Code 5 returns the XPath expression .//h1[@class="h2_entry"] for an extraction rule of <h2 class="h2_entry">. Code 6 has two function for extracting all results and the first result.

***Code 6.** Extraction with lxml*

```
from lxml import etree
from io import StringIO
def extract_all(html, pattern):
    parser = etree.HTMLParser()
    tree = etree.parse(StringIO(html), parser)
    result = etree.tostring(tree.getroot())
    root = tree.getroot()
    my_list = []
    yol = prepare_XPath(pattern)
    for elem in root.findall(yol):
        my_list.append(etree.tostring(elem,
        encoding='unicode'))
    return my_list
def extract(html, pattern):
    parser = etree.HTMLParser()
    tree = etree.parse(StringIO(html), parser)
    result = etree.tostring(tree.getroot())
    root = tree.getroot()
    elem = root.find(prepare_XPath(pattern))
    return etree.tostring(elem,
    encoding='unicode')
```

In Code 6, find method efficiently returns only the first match. findall method returns a list of matching Elements.

***Table 1.** Information about dataset*

Domain	Category	Avg. (KB)
Aliexpress	Shopping	93.66
Bild	Newspaper	67.49
Booking	Trip	689.11
Collinsdictionary	Dictionary	38.93
Ebay	Shopping	297.35
Imkb	Movie	214.23
Sciencedirect	Articles	48.65
Tchibo	Shopping	27.61
Tutorialspoint	Articles	28.56
W3schools	Articles	58.01
		153.36

4. Experiments

We prepare a dataset which contains web pages on many different content types, including scientific articles, dictionary, movies, newspaper articles, shopping, and trip/hotel information. For constructing this dataset, we have designed a simple crawler to download web pages. Then, this crawler downloads 30 web pages for every domain. Table 1 gives the average file size of web domains for this dataset. Moreover, we prepare extraction rules (like in Section 2) for every domain. All experiments are carried out on a computer using Intel Core i5-3.2Ghz processor and 8 GB RAM running Windows 10 operating system.

For measuring extraction time of these methods, we use time.clock method. This method returns wall-clock seconds elapsed, as a floating point number. This method is based on the Win32 function QueryPerformanceCounter.

4.1. Time results of Regex

There are two techniques in regular expressions. Table 2 indicates the extraction time results and whether the result of the extraction is correct or not.

***Table 2.** Time results and accuracy of regex*

Method	Avg. (ms)
extract	0.071
extract_all	0.307
	0.189
Accuracy: 390 / 600 = %43.5	

As expected, focusing only on one result rather than looking at the entire document has yielded better results. Extract method is 4.324 times faster than Extract_All method. However, 43.5% of the extraction rules give the correct result. In this case, DOM-based methods can be considered as a solution.

4.2. Time results of BeautifulSoup

BeautifulSoup supports two different extraction techniques and three different parsers for this task. Table 3 indicates the extraction time results.

***Table 3.** Time results of BeautifulSoup*

Method	Parser	Avg. (ms)
extract	html.parser	820.075
extract_all	html.parser	1192.317
extract	lxml	591.583
extract_all	lxml	1025.892
extract	html5lib	2191.472
extract_all	html5lib	2626.747
		1408.014

lxml parser for BeautifulSoup is the best parser for this task. html5lib and html.parser are just not very good results. As expected, the time results of extract methods are better than the time results of extract_all methods in all parsers. Finally, we examine lxml parser with its methods.

4.3. Time results of lxml

The lxml library is a binding for the C libraries libxml2 and libxslt. It is unique in that it combines the speed and XML feature completeness of these libraries with the simplicity of a native Python API. It can be used with BeautifulSoup, but it can be also utilized stand-alone. In this section, we examine this library stand-alone.

Table 4. *Time results of lxml*

Method	**Avg. (ms)**
extract	9.047
extract_all	9.480
	9.277

Time results of lxml is better than time results of BeautifulSoup. lxml parser in BeautifulSoup makes the results better, but the use of lxml stand-alone provides a much better improvement. (See Fig. 2 and 3 for comparing all libraries)

5. Conclusion

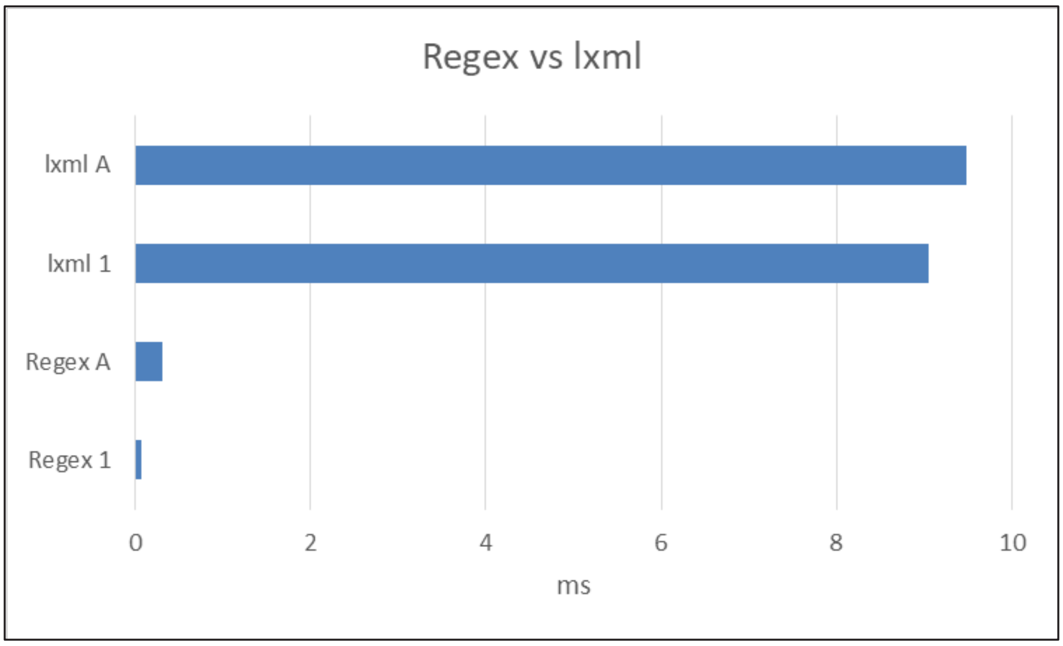
In this study, libraries of Python for extracting data from web pages are compared in order to understand their time durations. As expected, the experimental results show that regex gives better time duration with 0.071 ms. However, 43.5% of the extraction rules give the correct result. In this case, DOM-based libraries including BeautifulSoup and lxml can be considered as a solution. Lxml (stand-alone) provides better time results in DOM-based libraries. BeautifulSoup gave worse results because of extra processes for creating DOM even when using lxml parser.

In future work, we will need to develop more effective and effective methods for this task. Moreover, time results of other languages [11] on this task are compared.

Additional Information

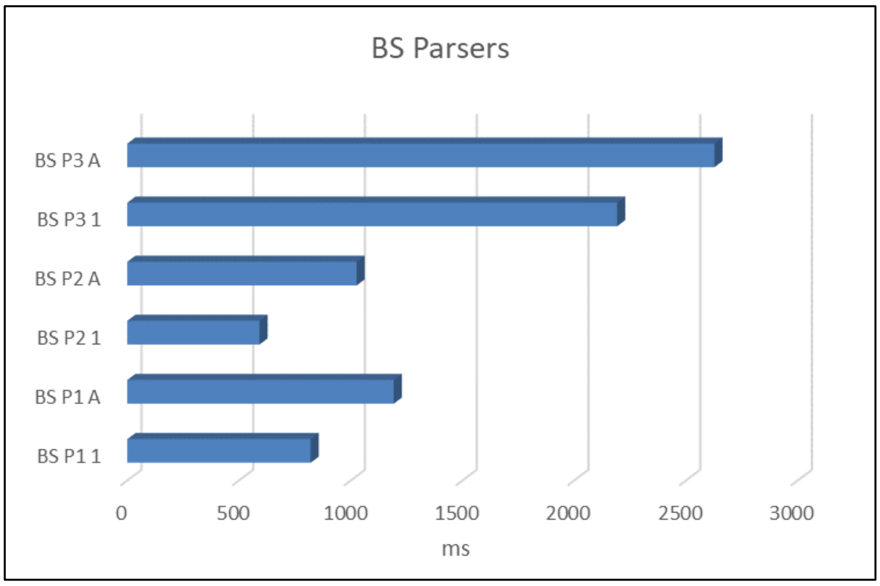
All codes are open-source. Source codes and dataset are as follows.

- https://github.com/erdincuzun/WebDataExtractionInPython



A: All extraction results, 1: only first result

Fig. 2. *Average time duration results of Regex and lxml*



A: All extraction results, 1: only first result, BS: BeautifulSoup, P1: html.parser, P2: lxml, P3: html5lib

Fig. 3. *Average time duration results BeautifulSoup Parsers*

REFERENCES

1. Rahman, A.F.R., Alam, H. and Hartono, R. (2001). Content extraction from HTML documents, *International workshop on Web document Analysis*, pp. 7-10, 2001.
2. Ferrara, E., De Meo, P., Fiumara, G., Baumgartner, R. (2014). Web data extraction, applications and techniques: A survey, *Knowledge-Based Systems*, Volume 70, 2014, pp. 301-323.
3. Flesca, S., Manco, G., Masciari, E., Rende, E., Tagarelli, A. (2004). Web wrapper induction: a brief survey", *In: AI Communications*, vol. 17, pp. 57–61. IOS Press, Amsterdam.
4. Álvarez-Sabucedo, L. M., Anido-Rifón, L. E. and Santos-Gago, J. M. (2009). Reusing web contents: a DOM approach, *Softw: Pract. Exper.*, 39: 299–314. doi:10.1002/spe.901.
5. Fu, L., Meng, Y., Xia Y. and Yu, H. (2010). Web Content Extraction based on Webpage Layout Analysis", Second International Conference on Information Technology and Computer Science, Kiev, 2010, pp. 40-43.
6. BeatifulSoup,https://www.crummy.com/software/BeautifulSoup/, (12.04.2018)
7. lxml, http://lxml.de/, (12.04.2018)
8. DOM, https://developer.mozilla.org/en-US/docs/Web/API/Document_Object_Model/Introduction, (12.04.2018)
9. Crummy,https://www.crummy.com/software/BeautifulSoup/bs4/doc/, (12.04.2018)
10. XPath, https://www.w3.org/TR/xpath/, (12.04.2018)
11. Uzun, E., Buluş, H. N., Doruk, A., Özhan, E. (2017). Evaluation of Hap, AngleSharp and HtmlDocument in web content extraction. *International Scientific Conference'2017 (UNITECH'17)*, Gabrovo, Bulgaria, November 17-18, Vol. II – pp. 275-278.

Authors' contacts

Erdinç Uzun,
Organization: Namık Kemal University, Çorlu Faculty of Engineering, Computer Engineering Department
Address: NKÜ Çorlu Mühendislik Fakültesi Dekanlığı, Silahtarağa Mahallesi Üniversite 1.Sokak, No:13, 59860 Çorlu / Tekirdağ / TURKEY
Phone (optional): +90 (282) 250 2325
E-mail: erdincuzun@nku.edu.tr

Tarık Yerlikaya (Corresponding author)
Oğuz Kırat
Organization: Trakya University, Faculty of Engineering, Computer Engineering Department
Address: Trakya Üniversitesi Ahmet Karadeniz Yerleşkesi Mühendislik Fakültesi 22020 Merkez / Edirne /TURKEY
Phone (optional): +90 (284) 226 1217 / 2215
E-mail: tarikyer@trakya.edu.tr, ogzkirat@gmail.com