WEB SCRAPING OR WEB CRAWLING:
A COMPILER - BASED APPROACH

A DISSERTATION

SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE AWARD OF DEGREE

OF

MASTER OF TECHNOLOGY

IN

SOFTWARE ENGINEERING

Submitted by:

FAHAD ABDULLAH

2K21/SWE/09

Under the supervision of

MS. SHWETA MEENA

DEPARTMENT OF SOFTWARE ENGINEERING

DELHI TECHNOLOGICAL UNIVERSITY

(Formerly Delhi College of Engineering) Bawana Road,

Delhi-110042

MAY, 2023

DELHI TECHNOLOGICAL UNIVERSITY

(Formerly Delhi College of Engineering)

Bawana Road, Delhi - 110042

CANDIDATE'S DECLARATION

I, Fahad Abdullah, Roll No. 2K21/SWE/09 student of M.Tech. (Software Engineering), hereby declare that the project dissertation titled **"WEB SCRAPING OR WEB CRAWLING : A COMPILER BASED APPROACH"** which is submitted by me to the Department of Software Engineering, Delhi Technological University, Delhi in partial fulfillment of the requirement for the award of the degree of Master of Technology, is original and not copied from any source without proper citation. This work has not previously formed the basis for the award of any Degree, Diploma Associateship, Fellowship or other similar title or recognition.

Place: Delhi

Date: May 2023

Fahad Abdullah

2K21/SWE/09

CERTIFICATE

I hereby certify that the Project Dissertation titled "**WEB SCRAPING OR WEB CRAWLING : A COMPILER BASED APPROACH**" which is submitted by Fahad Abdullah, 2K21/SWE/09, Department of Software Engineering, Delhi Technological University, Delhi in partial fulfillment of the requirement for the award of the degree of Master of Technology, is a record of the project work carried out by the students under my supervision. To the best of my knowledge, this work has not been submitted in part or full for any Degree or Diploma to this University or elsewhere.

Place: Delhi
Date: May 2023

Ms. Shweta Meena
Assistant Professor
Department of Software Engineering

ACKNOWLEDGMENT

The successful completion of this project required the assistance and input of a lot of people and the organization. I am grateful to all the people who helped me in creating the final result of this project report. I would like to express my sincere thankfulness to **Ms. Shweta Meena** for providing me with the opportunity to take up this project under their guidance. Without their sincere guidance and knowledge, this project would not at all have been possible. Their never-ending presence and motivation made me believe that the process of learning is much more important than the final outcome. I am also extremely thankful to the panel faculty during all the progress reports and intermediate evaluations for their directions, constant presence and for encouraging me to complete this project. They guided me through the whole process with new ideas, provided necessary information and pushed me to complete this project. I also thank all my fellow students for helping me in times when I needed it the most. It is because of their constant support that made me fulfill my academic responsibilities. Last, but not least I would like to thank my family for being there for me at times when no one is actually around. They have always tried to make it easier for me. There have also been numerous other people whom I cannot mention that have been a part of my life, directly or indirectly. I would like to thank them as well.

ABSTRACT

Data stored on the internet is tremendous and ever-increasing. Billions of users from around the world use the internet to search through a staggeringly huge number of web pages present on millions of websites over the web. The content over these web pages is both structured and unstructured making its analysis difficult as well as time-consuming. Web crawlers have emerged as a systematic and valuable tool for extracting data from these websites which helps businesses, researchers, and organizations to gather and understand huge amounts of data for various beneficial purposes. Because of the immense size of data on the web and the limitations in terms of bandwidth, storage, and time, everything that is present on the web cannot be analyzed due to which various approaches of web crawling and scraping became popular. Web crawlers are also aggressively used by search engines to index web pages and by businesses to increase their online presence. The thesis explores the field of web crawling and web scraping and mainly focuses on the use of the compiler-based approach of web crawling where a Python package like 'PLY' is proposed. The thesis further explores the advantages of this approach over the traditional web crawling approaches. Further, the widely used tools and strategies used in the field of web crawling are discussed. The thesis puts forward a clear overview of the areas where PLY excels over the traditional approaches. Furthermore, it also focuses on the applications in the real world where web crawling and web scraping is widely popular and in the end asserts the legal considerations related to this field.

CONTENTS

List of Figures

List of Tables

CHAPTER 1

INTRODUCTION

In the current digital era, web scraping and crawling have become indispensable strategies and in order for researchers, corporations, and people to fully utilize the potential of the internet, there are both possibilities and challenges presented by the massive amount of information that is already available. For several reasons, including market research, data analysis, sentiment analysis, and content aggregation, it is essential to extract data from websites. The process of gathering and organizing web data is complicated and tricky. Fig 1.1 shows a general flowchart explaining how data is fetched and extracted in general from the web using the web scraping techniques.

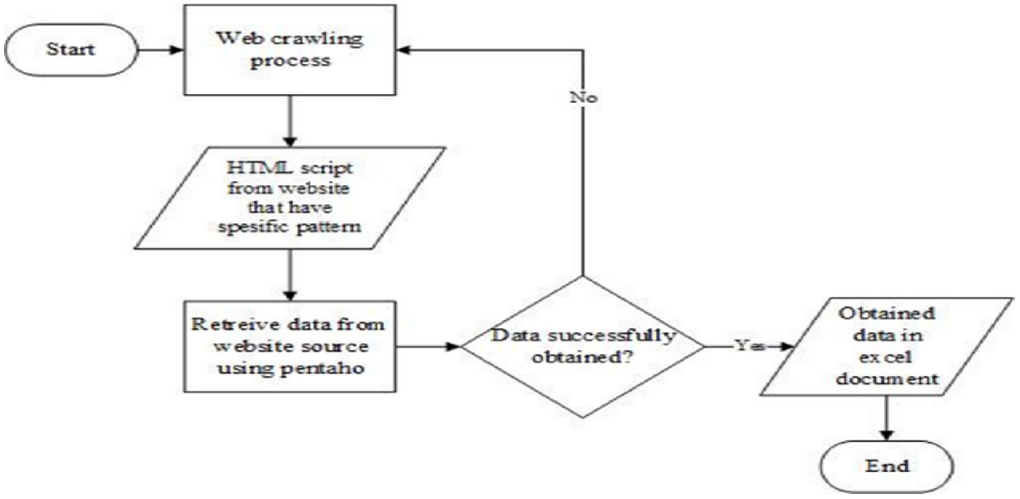


Fig. 1.1 General Web Scraping Flowchart [26]

Conventional methods of online scraping frequently rely on tools and libraries, like Beautiful Soup, that are primarily intended for parsing HTML or XML data. Despite their widespread usage and shown effectiveness in several situations, these technologies might not always be the most effective or flexible approach to collect data from the web. This is where the Python Lex-Yacc (PLY) package comes into play.

PLY provides a robust framework for building lexers and parsers, allowing programmers to define particular parsing rules and manage different document formats,

including non-standard markup languages or domain-specific languages. PLY provides fine-grained control over the parsing process, making it suited for a variety of scraping applications in contrast to Beautiful Soup, which depends on the predetermined structure and tags of HTML or XML documents.

In this thesis, we investigate the benefits of the PLY method over conventional web scraping methods. We explore the nuances of online scraping and crawling whilst discussing the challenges faced in extracting relevant information from the vast and diverse web ecosystem. Then, we present PLY as a versatile and effective tool that can get around some of the drawbacks of conventional methods.

The goal of the thesis is to emphasize the essential benefits and traits of PLY, demonstrating its capacity to manage complex web page structures, adapt to non-standard formats, and offer precise control over the scraping procedure.
Further, we also discuss the applications of web scraping in real world scenarios and discuss how it can be used for potential benefits and how businesses, researchers and developers can make use of this technology to bring value to the data-driven decision-making processes and research endeavors.

The thesis also discusses how web scraping is applied in real-world contexts, its potential advantages, and how companies, researchers, and developers can benefit from this technology to advance data-driven decision-making and research projects. The legality of online scraping is finally explored, along with the necessity of abiding by the rules while extracting data from the internet, terms of service, and respecting the rights of website owners. Data privacy, intellectual property rights, and the effects of internet scraping on performance are all discussed.

The purpose of this thesis is to thoroughly explain the benefits of utilizing PLY for web scraping and crawling to researchers, developers, and businesses. By embracing PLY's flexibility and power, new opportunities for gathering crucial data from the huge World Wide Web may be uncovered.

CHAPTER 2

RELATED WORK

Web crawling in today's world is most intensely used by search engines to index web pages and give out the search query results in the most relevant way possible. Google, the most widely used search engine in the world, claims to have indexed billions of online pages but largely keeps its indexing process a secret.. Search engines like Google, so big and advanced, also improves its crawling strategy day by day to have better search results and improved user satisfaction. These results are to be improved to avoid a major problem which search engines face these days, "search engine spamming" which is the exploitation of search engines by website owners to put their websites in the upper section of search results.

Research in the field improved the way web crawling is done and it is still done to refine the process even further. Researchers initially proposed the concepts of web mining on a general architecture [1] and that formed the basis of web crawling. The taxonomy proposed is still used to understand web crawling even today.

Web crawling mainly deals with huge amounts of data from web pages which is why big data also forms a significant basis in dealing with web crawling strategies. Big data analytics helps in obtaining quality results and the chances of its potential use in decision making also increases. Thus researchers, academicians and industries, also keep an eye on the technological developments in the big data domain to further improve the process of web crawling. The study [24] explains how web scraping is done in the era of big data and its usefulness for companies and governments.

Berendt's breakthrough proposal [2] describes semantic web as a major idea to add semantic annotations to web documents so that the information is accessed in a structured manner and managed in an automatic way. This not only helped in accessing the web information but also set forth methods as to how to make use of that information and manage it in a semi-automatic way. This brought forth the idea of web crawling and helped in identifying this area.

The use of deep learning methodologies to extract useful data from enormous amounts of unstructured data on the internet, helped in connecting the dots between big data and web crawling [3]. The idea of structural crawling of the internet as the data present on the web is enormous and unordered crawling may lead to confusion and no quality output [4].

The existing study conducted by researchers [5] helped in classifying web mining methodologies which in turn assisted in identifying the field of web crawling. The most important factor which put forth the technology of web crawling in the way it is perceived today is the development of search strategies for search engines. Search engines need to index the web pages on the web to give out the most useful and relevant outputs for the input search query or keyword. In the existing study [6] authors assisted in locating information of common interest from a large amount of available content which helped businesses bring out like-minded communities together and increase their markets, but somehow failed because information integration remained an unsatisfied property. A method is proposed for indexing the database population of web documents in the existing experiments [7]. These databases aid search engines in answering queries given by the user. Due to issues with information integration and the lack of a means to update and merge newly crawled links with existing ones, this technique also experienced a setback.

Generic crawlers as their name suggests crawled on different topics of various domains as mentioned in [8]. On the other hand, another study proposed focused crawlers for crawling on specific topics with knowledge beforehand[9]. Generic crawlers are however more useful in real-life applications.

Pre-identifying the query beforehand and then crawling it accordingly was another useful approach [10] which helped in obtaining specific results related to that query. The study done [11] analyzed how data repositories are to be built by web crawlers to be used for page indexing and better search outputs. A few other efficient approaches are proposed in [12][13] and [14] to crawl the world wide web.

APIs in recent years also gained importance to share application data and communicate easily amongst, but the problem with them is that not all of them are free to use and they provide only limited data. Thus dependency on APIs cannot guarantee quality data from the internet and this requires the use of web crawling to make sure that the data obtained matches business standards and needs [15][16].

Thus, the research in this area, initially began with identifying the significance of web crawling and furthered towards recognizing the importance of big data in this field. The methods utilized in this sector have also been enhanced by technological advancements in deep learning techniques. The results obtained got additional improvements by other techniques proposed such as structured crawling, generic crawling as well as focused crawling. It was also found that APIs cannot be relied upon to get quality results from web crawling.

CHAPTER 3

PROPOSED WORK

As the internet's size began to increase and the number of websites on it increased significantly, web crawling began to become a technology of immense importance. The method proposed in this thesis took inspiration from the compilers. The technique was thought upon while designing a compiler based calculator using Python [27]. The method was to read the lexems of the language token by token, try to identify them using regular expressions and then attempt to understand their context and take useful information out of it using grammars. Similar to compilers which read the high level language syntax and then try to understand the tokens using a grammar, the method discussed here tries to explain how web pages can also be parsed the similar way and read in a thoughtful manner. The grammar is designed such that the contents of the web page are understood accordingly.

To understand the basis of the proposed work, the explanation below shows how a compiler is made using python language to calculate basic math expressions. The following points shows how tokens are defined using python and how they are interpreted using regular expressions. The identification is done using a python package called ‘lex’. Later, these tokens are shown to be parsed using the yacc package.

3.1. Only ‘lex’ is used to demonstrate how tokens are identified

The program which is used to identify the tokens is called a Lexer and the module below shows how it is done. Step 1 is to import the lexer module from the PLY package. Then the tokens which are going to be present in the input string are identified and named. Next, the tokens are defined using regular expressions. For simple tokens like ‘+’, ‘-’, ‘*’, ‘/’, ‘%’, ‘(‘ and ‘)’ we interpret them simply by defining the symbol associated with them. For complex tokens like ‘INT’, ‘FLOAT’ we define them properly by identifying the regex(short for Regular Expression) associated with them. Like for defining ‘NAME’, we must identify that the variable name must contain alphabets, numbers and underscores but it must not start with a digit. We also define that

the empty spaces must be ignored and also that for "Illegal Characters" like '@' we must skip that token and go for the next token. Finally we create the lexer and then get the input and supply it to the lexer. We then print all the tokens of the lexer one by one using a loop.
The tokens in the lexer parser system are defined as:

```
''
tokens=['ADD','SUBTRACT']
''
```

And the regular expressions for these tokens are declared as :

```
''
t_ADD= r'\+'
t_SUBTRACT = r'\-'
t_ignore = r' '
''
```

All other math expressions can be shown the same way. Ignore is used to skip characters which are not required. Further, INT, NAME, FLOAT and error expressions are defined and when the input string is given to this lexer, the output is the recognition of the lex tokens defined.

For input:

```
''
data = 'a_123@ = 1 + (2 - 3)'
''
```

The output is shown as:

```
''
LexToken(NAME,'a_123',1,0)
Illegal Characters
```

```
LexToken(EQUALS,'=',1,7)
LexToken(INT,1,1,9)
LexToken(ADD,'+',1,11)
LexToken(LPAREN,'(',1,13)
LexToken(INT,2,1,14)
LexToken(SUBTRACT,'-',1,16)
LexToken(INT,3,1,18)
LexToken(RPAREN,')',1,19
‘’
```

In this way. the tokens of the input strings are identified.

3.2. Lex and Yacc both are used to read the input string token by token and parse the string only PLUS (addition) is shown

Further, to parse an expression, the parser is defined along with lexer. The grammar for the parser is used to evaluate expressions accordingly.
The function for calculating the entire input string is evaluated as:

```
‘’
def p_calc(p):
 ''' calc : expression |
            var_assign |
            empty '''
     print(run(p[1]))
‘’
```

Other functions are also defined along with their grammars so that the evaluation is done correctly.
To evaluate an expression, grammars are used as:

```
‘’
''' expression : expression PLUS expression '''
‘’’ var_assign : NAME EQUALS expression ‘’’
‘’’ expression : INT ‘’’
```

```
''' expression : NAME '''
''
```

The above grammars when used correctly while parsing evaluate a math expression correctly and assign the value of the expression to the variable. The output to the string:

```
''
" x = 2 + 3 "
''
```

is obtained as:

```
''
>> x = 2 + 3
" ,  x , (' + ' , 2 , 3 )
>> x
5
''
```

The program not only reads the input token but also evaluates the expression.

Now that the premise is set, the PLY package can be shown to work on web pages where data can be scraped using this package with the help of grammars based on the context of the page.

The actual working of the scraper can be shown as:

- First, the 'urllib' package helps in obtaining the contents of the webpage. The contents are first refined and decoded to fit a specific requirement.

```
'' import urllib.request ''
```

- Other packages like ‘re’ and PLY are then used to understand and extract the contents of the webpage and then take out various details of the movie entered by the user.

```
‘’ import re
  import ply.lex as lex
  import ply.yacc as yacc ‘’
```

- Urllib package crawls to the url given as input.

```
‘’ response = urllib.request.urlopen(url)
  webContent  = response.read()
  webContent = webContent.decode(‘utf-8’) ‘’
```

- Re package helps in identifying regular expressions to help lexer(lex) identify tokens appropriately. Like the “MNAME” token is used to identify the movie name.

```
‘’ tokens = [“MNAME”]
<h1\sslot="title"\sclass="scoreboard__title"\sdat
a-qa="score-panel-movie-title">(.*?)</h1> ‘’
```

- Parser (yacc) generates the parse tree internally using grammar rules and extracts the information required by the user related to a particular movie.

  The grammar used in the parser is shown as:

```
‘’
‘’’ moviename : MNAME |
              empty ‘’’
‘’
```

- In this way, with suitable grammar a web page can be extracted using only the lexer and parser approach.

Extracting name from a large unstructured HTML webpage:

- Web Content is scraped and regex is defined precisely to extract particularly the movie name from a large unstructured HTML page.
- Parser then gives suitable grammar to save the movie name in log_movie dictionary.

This is just a short example of extracting only the name from a set of other values which could be extracted using the same method and with appropriate grammar rules for each value extracted.

The output of the above code is shown as:

```
(base) dell-lmac-41980:Python Calculator using ply Fahad1$ python 6-movie
nameParser.py
LexToken(MNAME,'<h1 slot="title" class="scoreboard__title" data-qa="score
-panel-movie-title">Black Panther</h1>',3324,255243)
Black Panther
(base) dell-lmac-41980:Python Calculator using ply Fahad1$
```

Fig. 3.1 Output to extract movie name from html web content using ply

As seen clearly, the Lex Tokens for movie_name are accurately fetched and identified and the parsing of movie name is done. It shows the Movie Name correctly as shown in the above output.

The grammar to evaluate movie name from the html web content is shown as :

```
start -> moviename
moviename -> MNAME | empty
empty ->
```

The parse tree is shown as:

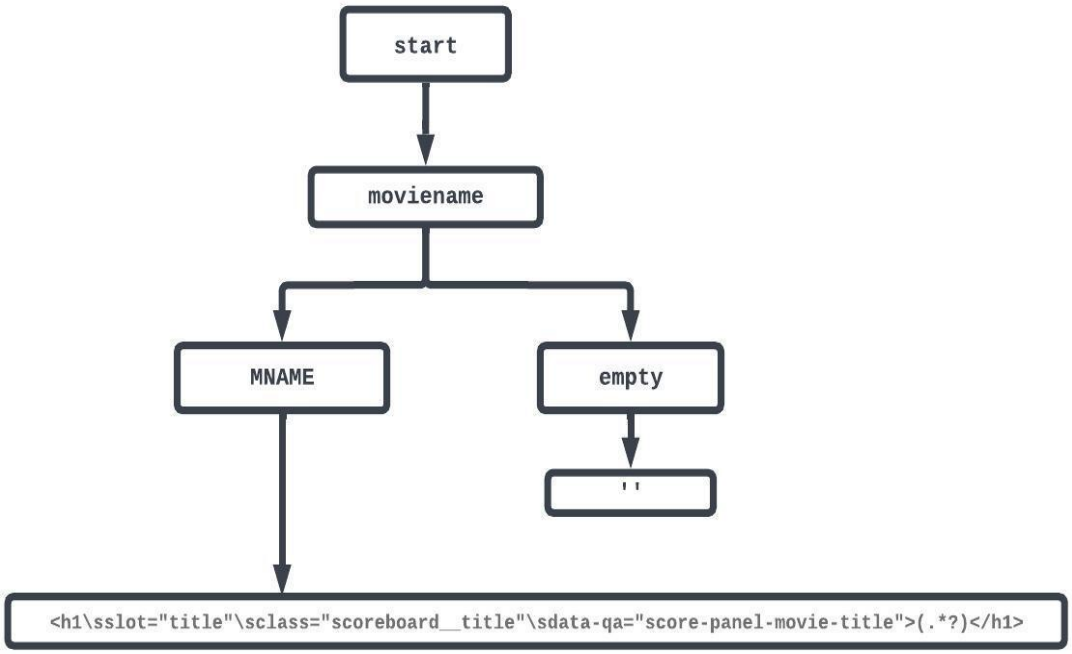


Fig. 3.2 Parse tree to evaluate movie name from html content

The parse tree evaluates the html web content based on the grammar shown above. The exact grammar is used in the code for the above extraction.

The last block in the above diagram shows the regular expression based on which the movie name is identified.
Other details about the movie name can also be extracted and the complexity of the grammar as well as parse tree is going to increase accordingly.

CHAPTER 4

WORKING AND ANALYSIS

The PLY methodology is an effective method for parsing and information extraction from structured text, such as web pages. Lexical analysis and parsing are the two steps involved. The input text is divided into a series of tokens, which stand in for meaningful units like words, numerals, and symbols throughout the lexical analysis step. After analyzing the token sequence using a set of grammatical rules, the parsing stage creates a hierarchical structure that captures the connections and semantics of the input. The PLY technique provides efficient and reliable extraction of necessary information from complicated web pages by setting the proper lexer and parser rules.

4.1. WORKING

The exact working of the above approach to extract the useful details from the given html contents of a webpage can be explained in the following steps:

I. Importing libraries is the first important and necessary step and modules like ply.lex and ply.yacc are imported to be used for lexical analysis and parsing respectively. The ‘urllib’ library is used to fetch the HTML content of the webpage. Package like ‘re’ is used to match the given string with a given regular expression which is used to recognise the information given in the webpage.

II. Next, the URL is specified and the HTML content is fetched from the given URL using the request functionality of the ‘urllib’ package.

III. Now that the content is available and decoded to fit the requirements of lexical and parsing stages, the next step is to specify the lexer and parser rules.

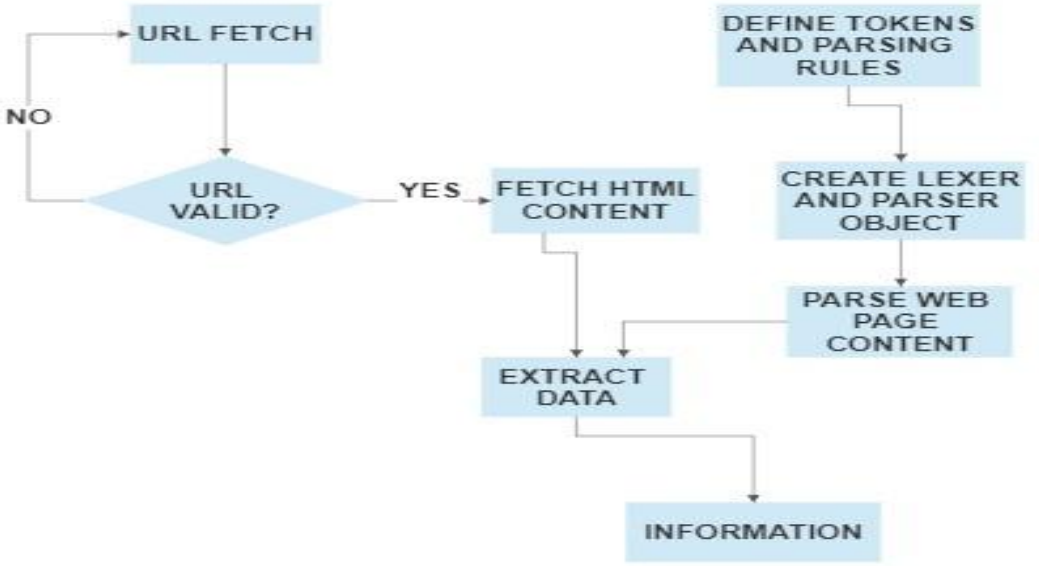


Fig 4.1 Working of PLY package flowchart

IV. Lexer Tokens are defined which are going to be used in the lexical analysis phase. Then, the lexer rules are defined to fetch different types of information, along with the appropriate lexer rules for newline, error and ignore cases.

V. Now, parser rules are defined to extract the exact information. Most importantly, correct and well-thought grammar rules are also important to be defined with the parser rules so that the correct parser tree is built internally without any ambiguity.

VI. After all the necessary steps are done for the lexer and parser, both can now be built using the lex.lex() and yacc.yacc() function. This builds the lexer and the parser instance with the defined token rules, parser rules and error handling functions.

VII. Now that everything is set up, the web content extracted above can be given to the parser which will internally create the LALR tables and then the parse tree to give out all the requested information from a given webpage.

The above steps show the working of the PLY based approach to extract the information from a given webpage.

The code shown above only shows the extracted movie name but the actual working version has a series of options which are given to the user to select from. The

information extracted is obtained using a series of well defined and well-thought grammar rules which are defined in the parser.

4.2. ANALYSIS

Although the PLY or the compiler-based method of online scraping is undoubtedly different from the conventional methods; in some cases it is more effective and useful.

The PLY technique, or compiler-based approach, offers customized parsing rules that handle the parsing of complex web pages more adeptly than Beautiful Soup. Besides that, employing beautiful soup can be difficult when dealing with complex data structures that have intricate and nested configuration. PLY, on the other hand, offers personalized parsing rules that are effective at handling complex data structures. For instance, PLY makes it simple to extract specific data from each table cell on a website with nested tables by allowing users to build parsers that can swiftly navigate through the layered structure and find the required data efficiently.

PLY has additional applications for non-standard markup languages. On the one hand, because HTML and XML documents are by definition markups, the beautiful soup and the scrappy techniques are largely focused on parsing such documents. However, those approaches have difficulty parsing websites that utilize non-standard markup languages or domain-specific languages. Due to its flexibility to create customized parsers, PLY thrives in such situations and makes it possible to effectively extract information from non-standard websites.

Conventional methods heavily rely on tags and attributes making them ideal only for those scenarios where the content is available in that form. Situations where target data is unstructured or semi-structured are handled using PLY with ease.

PLY proves to be even more effective when integrated with external tools like Selenium or PhantomJS which allows the scraping of dynamically generated or JavaScript-based content.

Another major advantage coming to PLY is that there are packages similar to PLY in other programming languages which offer similar functionality to build custom lexers and parsers. While PLY is specific to python, tools like ANTLR(Another Tool for Language Recognition) which supports languages like Java, C# and JavaScript allows the user to define grammar and generate lexers and parsers, Jison(JavaScript-based parsing toolkit uses Bison and Yacc), JavaCC or Java Compiler Compiler (another popular parsing tool for Java applications) and Lex and Yacc(classical tools for parsing in c and C++) are available which can help in building the parsers similar to PLY and thus can help in scraping the web using programming languages other than Python. This proves the language independence of this compiler-based solution allowing the implementation of this solution across different programming languages. This allows specific advantage of this approach making it not only portable but also increases its consistency and ecosystem integration power.

As a result, the solution for online scraping works extremely well for unstructured or semi-structured data, allowing for the creation of unique parsing rules, provides fine-grained control over complex parsing scenarios, and proves to be language independent and portability. This approach accommodates dynamically created data if correctly integrated with external tools, making it very effective and practical in some situations when existing traditional methods are ineffective.

CHAPTER 5

GENERAL TOOLS AND STRATEGIES

This section explores several methods for obtaining data from web sites. These include using regular expressions to look for certain patterns in text, using parsing libraries to traverse and extract data from HTML or XML documents, and using sophisticated parsing techniques to extract data accurately and efficiently. It also examines several web scraping techniques and algorithms. By understanding and employing these tools and strategies, developers can effectively and efficiently extract the desired data from websites

5.1. TOOLS

Web Crawling can be done by using methods that read the web document by downloading it or using some packages which already contain all the methods used for web crawling. It can be done on HTML and XML web pages. The tools used in web crawling are explained below-

5.1.1 Beautiful Soup: It is a package in python language which is used to extract useful data in a structured form from an unstructured web page. It uses HTML or XML parsers to parse web pages and convert them into complex python object trees. Easy in its use, it can be used to navigate and examine a parse tree in a few steps. It also automatically converts the web page into compatible encodings to parse them faster and thus there is no need to convert the document encoding by the user.

*5.1.2. Regular Expression***:** Python provides a package called ‘re’ to work with regular expressions to identify patterns in the webpage and match them according to our requirements. These regular expressions allow the user to work with any web page and they form the basis of any form of web crawling and scraping. A regular expression is specified matching the pattern which is to be found on the web page. The example given below returns a list of numbers present in the given string.

Input:

```
string = """My name is Regular Expression.
My phone number is 912471374123 and my landline number is
4834873"""
# A sample regular expression to find digits.
regex = '\d+'
match = re.findall(regex, string)
print(match)
```

Output:

```
['912471374123', '4834873']
```

5.1.3. LXML: Python provides lxml to parse XML and HTML pages. It is easy to learn and is fastest and rich in features. Its installation is moderately difficult but when it comes to its ease of use, it is the best parser. It is used widely in the industry because of its power but the only drawback is its difficult installation.

These three methods are used to exploit their trade-offs according to the user's needs. Beautiful soup, on one hand can be used to retrieve data from the website if the constraint lies in downloading the web pages. It is a pure python approach that is moderately easy to install and use and faster in generating results. Regular expressions can be easily used to get limited data from the web page but have almost no dependency on any module/package to work with except for the package which supports regular expressions like the 're' package in python. Lxml is by far the best and the most popular approach[17]. It is fast and potent from all aspects while other approaches are limited to specific settings.

5.2. STRATEGIES

Web crawling starts by giving the algorithm a seed URL that is visited first. The web page for the seed URL is parsed and links are extracted from it. These links are then visited algorithmically according to the scheduling policy defined by the search engine. As the size of the web is huge, not all the links are visited but only the most relevant

ones. The relevance of these pages is dependent on search engine methodologies of indexing. Better the seed URL, the better the search results. The strategy using which these web pages are downloaded also determines the quality of the output produced. Some of the most popular web crawling strategies as mentioned in [20] [21] are:

5.2.1. Breadth-First Search:

BFS goes from the seed URL to all neighbor links in the next level. It goes down level by level until the required URL is found or else accounted as a failure. For searches in which the desired URL is expected to be found closer to the seed URL, BFS is suited.

5.2.2. Depth First Search:

DFS traverses the URLs in such a manner that it first visits all the child links from one page, then visits all links extracted from this child page, i.e. children of this child URL. This way instead of a shallow search it searches deeper. For larger child pages, DFS fails and remains unterminated [18].

5.2.3. Page Rank Algorithm:

Backlinks and citations are considered an important factor while indexing web pages by web crawlers. These factors are specifically taken into account while writing page ranking algorithms. These concepts are also exploited and misused sometimes in “search engine spamming”. In a specific web page, consider only those input links which contribute more to a search query.

5.2.4. Largest sites crawling:

A study done[18] shows that crawling larger sites first is a highly useful strategy as it often gives the most relevant links in a very less time. The crawling algorithm first finds all the highest priority pages and then visits the largest pages first in this strategy.

5.2.5. Importance calculation for the web page:

To find the importance of a web page, a parameter called cash value is used. Cash value is dependent on the output links present on a web page.More the output links, more the importance of a page. This method, however, downloads every page multiple times which complicates the crawling process and is also time taking.

5.2.6. Focussed Crawling method:

Focussed crawling works by assigning similarities to web pages that are identical to a given search query. This approach first checks the relevance of a web page for a given query by various similarity indexes and then decides whether to download the web page or not. The content of already visited pages also acts as a deciding factor for getting to a new linked page [19].

CHAPTER 6

APPLICATIONS AND LEGALITY

The application and legalities section discusses the practical applications of web scraping in various domains and industries.It examines how data is gathered through online scraping for things like market research, competitor analysis, content aggregation, and data analysis. The section also discusses the legal implications and issues of online scraping, such as the significance of abiding by website terms of service, comprehending intellectual property and copyright laws, and being aware of data privacy laws. This section offers insights into the advantages and moral issues connected with this effective data collecting approach by addressing both the uses and legality of web scraping.

6.1. APPLICATIONS

Technological advancements are rapidly evolving and improving in today's world. A huge repository of information present on the world wide web attracts users belonging to various fields like research, academics, and business intelligence, from around the world to gather the information and make use of it in a manner useful to them. Web scraping and crawling assist by making the overall web understandable to them. Relevance indexes, page indexes, and other approaches help users download the web pages according to their needs and relevances, understand them with the help of scraping and crawling methods, and then jump from one URL to another by extracting links. This helps the users to bypass the problems like storage restrictions, computation capability restrictions, and time crunches that may be encountered because of the enormous web size. Web scraping techniques also come in handy with big data specialists for collecting and analyzing huge amounts of data in ways that were previously unimaginable. Automatic data collection methods help big data specialists save a lot of time to examine large numbers of web pages and the data present on them.

Because of the tremendous amounts of data generated daily on the internet, these techniques are extensively acknowledged and recognized as potent and effective in

collecting that data and understanding it. Many manual, ad hoc, and automatic techniques are identified and accepted in today's world to capture complete web pages into structured databases and these techniques are improving day by day.

Web scraping methodologies are widely recognized and used in areas like -

- Big Data
- Data Mining
- Cyber Security
- Marketing
- Cloud Computing
- Research
- Business Intelligence
- Business Competition
- Personal tools

6.2. LEGALITY

Researchers are proposing new methods and new tools are designed every now and then for web scraping but the legality to utilize these tools is a gray area that is often overlooked and ignored. Various ethical principles are designed for web scraping to disallow misconducts like “breach of contract”, “copyright infringement”, “damage to a website”, etc. but specific details about the legal and ethical frameworks is still an area that needs to be addressed by the courts. The legality issues as mentioned in [22] and [23] are provided below.

A. Any website owner can legally prohibit programmatic access to their website by specifically disallowing this in the “terms of use” policies on the website and any programmer who tries to access such a website results in a “breach of contract”.

B. Using some websites' images, pictures, or data without adequate access rights, scraping and publishing it as one’s own may lead to copyright infringement. Copyrighted material can be used only under the fair use principle.

C. It is also protected by law to make use of scraped data in some illegal or fraudulent activities.

D. "Trespass to chattels" law guarantees the prosecution of users if the web scraping used by them on a website leads to any type of damage to the website or server on which the website is hosted, and it is proven in the court. It guarantees financial compensation to the website owner, depending on the intensity of the damage done.

Fig 6.1 Legality and Ethics Framework for Web Scraping [22]

Overall, the legality of web scraping tools and methods is still an area that is to be looked into by the courts to guarantee the safeguarding of content and data of the owner of the website in a way the owner wants. If the owner allows the website to be open and scraped, then it can allow programmatic access to their website while if the owner does not allow it, it is to be mentioned in the terms of use and followed strictly by web scraping programmers and faulty considerations of these ethics must be guaranteed lawsuits by the court of law.

CHAPTER 7

RESULTS

The purpose of this thesis was to explore the compiler-based approach of web scraping and compare it with other conventional approaches like Beautiful Soup and Scrapy. The thesis highlights the advantages of the PLY - based approach and provides detailed comparison between them. It also explains the scenarios where the conventional methods find it difficult to perform the task and the helpfulness of compiler-based approach comes into play.

PLY offers several advantages which are explained below:

I. PLY-based approach allows developers to define their parsing rules using grammar specifications.Complex parsing tasks involving intricate parsing scenarios can be handled using PLY. It also provides fine-grained control over the scraping process making the parsing of large unstructured html pages seamless.

II. Non-markup and domain-specific languages can also be handled by PLY easily which the conventional approaches fail to handle. Traditional methods parse HTML and XML formats because they look for specific tags but when it comes to non-standard languages, these approaches fail and due to the custom parsing capabilities of PLY approach, these cases are easy for it to handle.

III. PLY seamlessly integrates with Selenium and can help in scraping dynamic web content and thus this approach seems to work with external tools in web pages which require JavaScript rendering.

IV. PLY relies on grammar rules, making it low level and customizable which offers more flexibility and control while beautiful soup mainly relies on the structure of HTML and XML documents to navigate and extract data.

V. The method discussed is not restricted to one programming language and almost all the major programming languages have some compiler toolkit which can help in parsing web pages. This shows the language independence of this method and the approach is not restricted to Python programming language.

The following table summarizes the differences between the approach proposed and the approaches available:

Table 7.1 Comparison between traditional and proposed approach

Comparison Factors	PLY-based approach	Traditional Approaches
Parsing Method	LR Parsing	HTML Parsing
Flexibility	High	Medium
JavaScript Execution	Seamless Integration with Selenium	Limited Support
Scalability	High	Medium
Error Handling	Customizable	Basic
Ease of Use	Moderate	High
Performance	Efficient	Moderate
Automatic Crawling	No	Yes
Extensibility	High	Medium
Language Independence	Yes	No
Customization Capabilities	Extensive	Limited
Integration with Existing Tools	Versatile	Limited
Additional Frameworks	No	Yes

Thus the PLY-based technique for online scraping has a number of benefits. It is a potent tool for extracting data from web pages because of its adaptability, language independence, effective parsing, and connection with Selenium. The comparison of this

approach and other approaches emphasizes the advantages and disadvantages of each strategy, enabling developers to select the best solution for their particular scraping requirements. Web scraping projects may be made more successful and efficient by developers by utilizing the advantages of the PLY - based methodology.

CHAPTER 8

CONCLUSION

The thesis initially introduced web crawling and scraping in general and then dived into an unconventional approach to web scraping.

It investigated the usage of the PLY based approaches as an effective strategy for web crawling and scraping and focused on the benefits of using PLY instead of more conventional techniques, such as Beautiful Soup, for collecting data from websites.

The PLY approach's use and analysis throughout the thesis demonstrated its effectiveness, adaptability, and language independence. PLY's integration with Selenium and other technologies has proven its capacity to manage dynamic information and navigate complex web structures. Developers and academics may quickly and easily extract structured data from webpages by utilizing PLY's robust parsing capabilities in specific scenarios.

The thesis also covered the legal and moral issues surrounding web scraping, emphasizing how crucial it is to abide by website terms of service and ask for permission when required. Understanding these aspects is crucial for conducting responsible and compliant web scraping activities.

REFERENCES

[1] R. Cooley, B. Mobasher and J. Srivastava, "Web mining: information and pattern discovery on the World Wide Web," in *Proceedings of Ninth IEEE International Conference on Tools with Artificial Intelligence*, 1997, pp. 558-567.

[2] B. Berendt, A. Hotho, D. Mladenic D, M. V. Someren, M. Spiliopoulou, and G. Stumme, "A roadmap for web mining: From web to semantic web," *European Web Mining Forum*, pp. 1-22, Springer, Berlin, Heidelberg, 2013 Sept. 22.

[3] M. M. Najafabadi, F. Villanustre, T. M. Khoshgoftaar N. Seliya, Wald, and E. Muharemagic, "Deep Learning Applications and Challenges in Big Data Analytics," *Journal of Big Data*, vol. 2, no. 1, pp. 1-21, Feb. 2015.

[4] G. David, K. Jon, and R. Prabhakar, "Inferring web communities from link topology," in *Proceedings of the 9th ACM Conference on Hypertext and Hypermedia: Links, Objects, Time and Space-Structure in Hypermedia Systems*, pp. 225–234, Pittsburgh, Pa, USA, 1998.

[5] R. Kosala, and H. Blockeel, Web mining research: A survey," *ACM Sigkdd Explorations Newsletter*, vol. 2, no. 1, pp. 1-15.

[6] A. Singh, M. Srivatsa, L. Liu, and T. Miller, "Apoidea: a decentralized peer-to-peer architecture for crawling the world wide web," in *Distributed Multimedia Information Retrieval: Proceedings of the SIGIR 2003 Workshop on Distributed Information Retrieval, Toronto,* vol. 2924, pp. 126-142, August, 2003.

[7] S. Pandey, and C. Olston, "User-centric web crawling," in *Proceedings of the 14th International Conference on World Wide Web*, pp. 401–411, 2005.

[8] A. Arasu, J. Cho, H. Garcia-Molina, A. Paepcke, and S. Raghavan, "Searching the web," *ACM Transactions on Internet Technology*, vol. 1, no. 1, pp. 2–43, 2001.

[9] F. Menczer, G. Pant, P. Srinivasan, and M. E. Ruiz, "Evaluating topic-driven web crawlers," in *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 241–249, 2001.

[10] S. D. Ramalingam, and D. Manjula, "Survey on comparative analysis of queries over historical time series," *International Journal of Computer Applications*, vol. 106, no. 6, pp. 34–37, 2014.

[11] S. Balamurugan, N. Rajkumar, and J. Preethi, "Design and implementation of a new model web crawler with enhanced reliability," in *Proceedings of the World Academy of Science, Engineering and Technology*, vol. 32, Aug. 2008.

[12] A. Z. Broder, M. Najork, and J. L. Wiener, “Efficient URL caching for World Wide Web crawling,” in *Proceedings of the 12th International Conference on World Wide Web*, pp. 679–689, May 2003.

[13] J. Cho, H. Garcia-Molina, and L. Page, “Efficient crawling through URL ordering,” in Computer Networks and ISDN Systems, vol. 30, no. 1–7, pp. 161–172, 1998.

[14] S. Chakrabarti, “*Mining the Web: Discovering Knowledge from Hypertext Data,”* Morgan Kaufmann Publishers, 2002.

[15] R. Mitchell, “Web scraping with Python: Collecting more data from the modern web,” O'Reilly Media, Inc., 2018 Mar. 2021.

[16] S. V. Broucke, and B. Baesens, “Practical Web Scraping for Data Science: Best Practices and Examples with Python,” Apress, 2018.

[17] R. Lawson, “Web Scraping with Python,” Packt Publishing Ltd., 2015.

[18] C. Castillo, M. Mauricio, and A. Rodriguez, “Scheduling Algorithms for Web Crawling,” in *Proceedings of WebMedia and Latin America Web Conference*, Preto, Brazil. 2004.

[19] S. J. Kim, and S. H. Lee, “An improved computation of the PageRank algorithm,” in *Proceedings of the European Conference on Information Retrieval,* pp. 73-85, 2002.

[20] R. Kumar, A. Jain, and C. Agrawal, “Survey of Web Crawling Algorithms” in *Advances in Vision Computing: An International Journal (AVC),* vol. 1, no. 2/3, September 2014.

[21] P. Dahiwale, R. Janbandhu, and M. M. Raghuwanshi, “Analysis of web crawling algorithms”, in *International Journal on Recent and Innovation Trends in Computing and Communication,* vol. 2, pp. 488 – 492, March 2014.

[22] V. Krotov, and L. Silva, “Legality and Ethics of Web Scraping,” 2018.

[23] M. Khder, “Web Scraping or Web Crawling: State of Art, Techniques, Approaches and Application”, in *International Journal of Advances in Soft Computing and its Applications*, vol. 13, pp. 145-168, November 2021.

[24] J. Snell, and N. Menaldo, “Web scraping in an era of big data 2.0,” *Bloomberg Law News*, 2016.

[25] Cooper, K.D. and Torczon, L. (2023) ‘Parsers’, in *Engineering a compiler*, 3rd ed. Cambridge, MA: Morgan Kaufmann Publishers, ch. 3, pp 91 - 94.

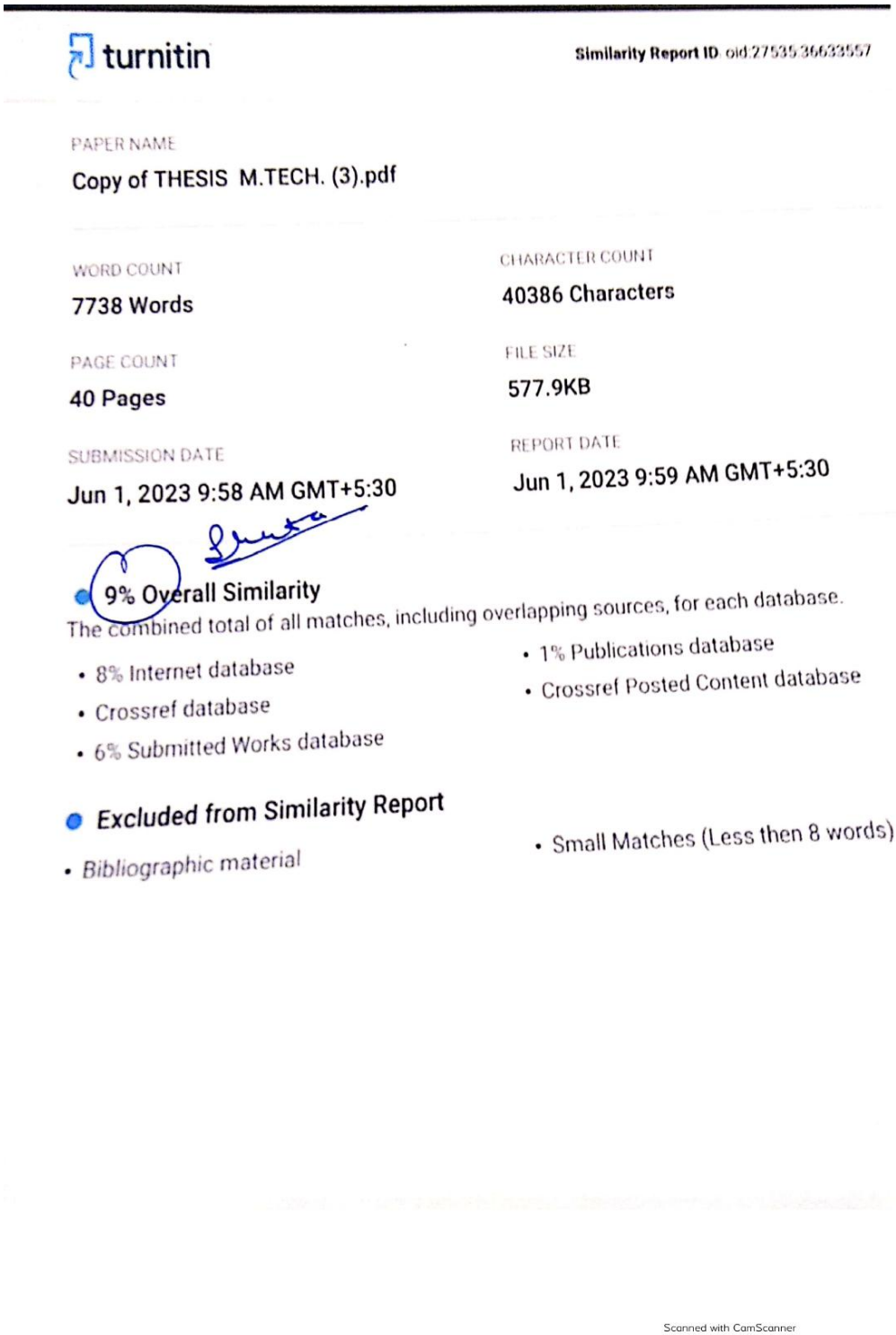
[26]L. R. Julian and F. Natalia, "The use of web scraping in computer parts and assembly price comparison," 2015 *3rd International Conference on New Media*

(CONMEDIA), Tangerang, Indonesia, 2015, pp. 1-6, doi: 10.1109/CONMEDIA.2015.7449152.

[27]"Making your Own Calculator in Python," *YouTube*. http://www.youtube.com/playlist?list=PLBOh8f9FoHHg7Ed_4yKhIbq4lIJAlonn8

APPENDICES

Plagiarism Report



turnitin

Similarity Report ID: oid:27535:36633557

PAPER NAME
Copy of THESIS M.TECH. (3).pdf

WORD COUNT
7738 Words

CHARACTER COUNT
40386 Characters

PAGE COUNT
40 Pages

FILE SIZE
577.9KB

SUBMISSION DATE
Jun 1, 2023 9:58 AM GMT+5:30

REPORT DATE
Jun 1, 2023 9:59 AM GMT+5:30

9% Overall Similarity
The combined total of all matches, including overlapping sources, for each database.

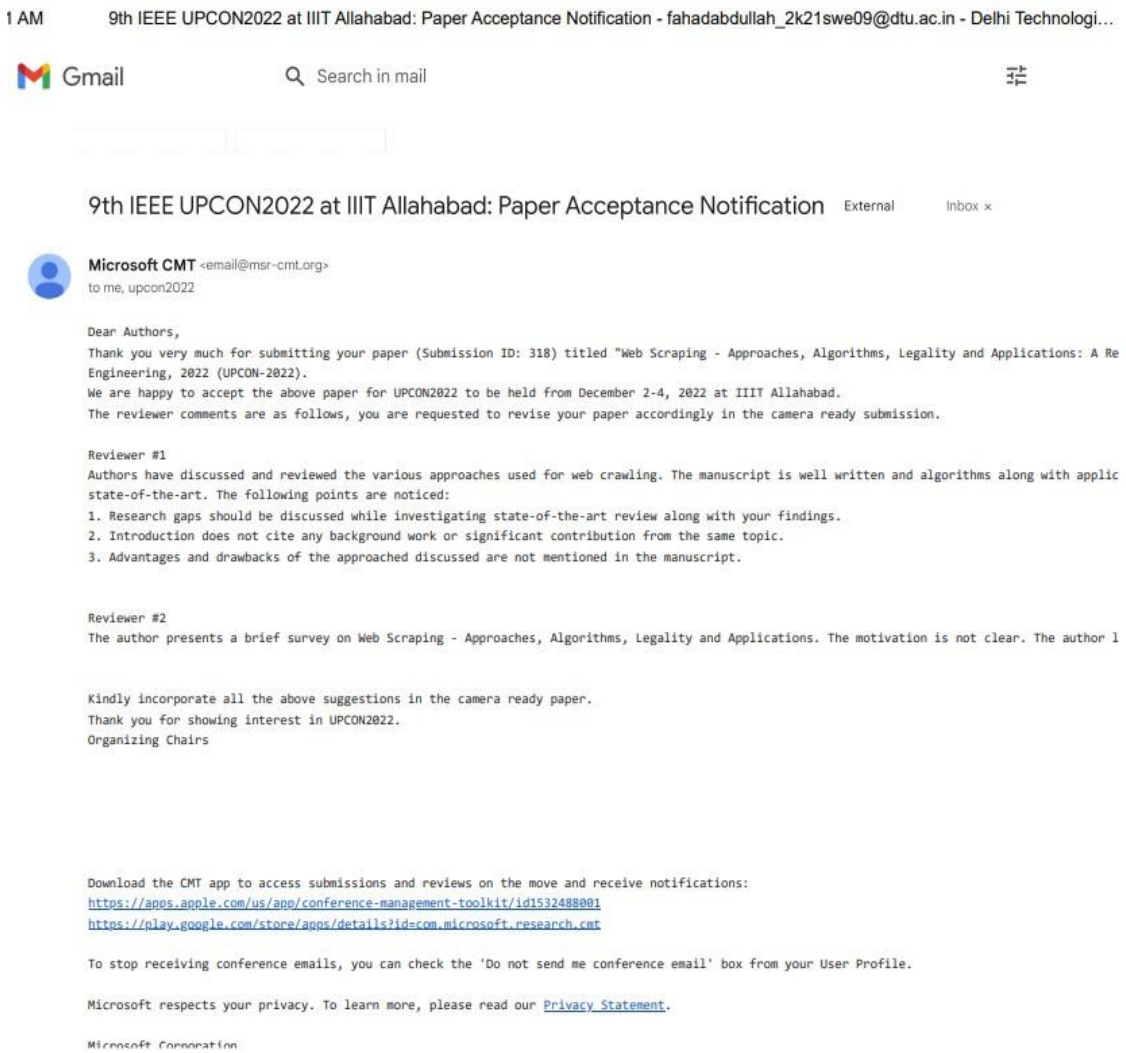
- 8% Internet database
- Crossref database
- 6% Submitted Works database
- 1% Publications database
- Crossref Posted Content database

Excluded from Similarity Report

- Bibliographic material
- Small Matches (Less then 8 words)

Scanned with CamScanner

Paper Acceptance Mail



1 AM 9th IEEE UPCON2022 at IIIT Allahabad: Paper Acceptance Notification - fahadabdullah_2k21swe09@dtu.ac.in - Delhi Technologi...

Gmail Search in mail

9th IEEE UPCON2022 at IIIT Allahabad: Paper Acceptance Notification External Inbox ×

Microsoft CMT <email@msr-cmt.org>
to me, upcon2022

Dear Authors,
Thank you very much for submitting your paper (Submission ID: 318) titled "Web Scraping - Approaches, Algorithms, Legality and Applications: A Re
Engineering, 2022 (UPCON-2022).
We are happy to accept the above paper for UPCON2022 to be held from December 2-4, 2022 at IIIT Allahabad.
The reviewer comments are as follows, you are requested to revise your paper accordingly in the camera ready submission.

Reviewer #1
Authors have discussed and reviewed the various approaches used for web crawling. The manuscript is well written and algorithms along with applic
state-of-the-art. The following points are noticed:
1. Research gaps should be discussed while investigating state-of-the-art review along with your findings.
2. Introduction does not cite any background work or significant contribution from the same topic.
3. Advantages and drawbacks of the approached discussed are not mentioned in the manuscript.

Reviewer #2
The author presents a brief survey on Web Scraping - Approaches, Algorithms, Legality and Applications. The motivation is not clear. The author l

Kindly incorporate all the above suggestions in the camera ready paper.
Thank you for showing interest in UPCON2022.
Organizing Chairs

Download the CMT app to access submissions and reviews on the move and receive notifications:
https://apps.apple.com/us/app/conference-management-toolkit/id1532488001
https://play.google.com/store/apps/details?id=com.microsoft.research.cmt

To stop receiving conference emails, you can check the 'Do not send me conference email' box from your User Profile.

Microsoft respects your privacy. To learn more, please read our Privacy Statement.

Microsoft Corporation

Scopus Index Proof