UNIVERSITY OF WATERLOO

Faculty of Science

Automated Raw Component Parameter Information Extraction: Algorithms and Methods Comparison

Tata Consultancy Services
International Tech Park,
Bengaluru, India

Satvik Varshney
ID 20764052
3B Honours Science and Business, Physics

September 15, 2021

September 15, 2021

Dr. Kashif Memon, Advisor, Honours Science and Business Program Faculty of Science
University of Waterloo
200 University Avenue West

Waterloo, Ontario N2L 3G1

Dear Dr. Memon:

This report, titled "Automated Raw Component Parameter Information Extraction: Algorithms and Methods Comparison" was prepared as my Spring 2021 Work Term Report for Tata Consultancy Services. This is my third work term report.

Tata Consultancy Services (TCS) is an Indian multinational IT services and consulting firm located in Mumbai, Maharashtra. I was employed under the Supply Chain department of TCS working directly for a major San Jose based communications company, under the mentorship of Mr Dilip Kumar (Supply chain program manager).

The purpose of this report is to investigate and compare the developed methods for automating the process of parameter extraction of raw components used by the customer's "Component and Technology" department in their final products. These raw components are then to be listed in their approved Bill of Materials. The automation is developed to reduce time spent by skilled engineers on mundane tasks like obtaining the component parameters, and allows them to focus on the higher value tasks at hand.

This report was written entirely by me and has not received any previous academic credit at this or any other institution. I would like to thank Mr Dilip Kumar, and Ms Snigdha Singh for providing me with valuable support and data resources used in the preparation of this report. I received no other assistance

Sincerely,
Satvik Varshney
ID 20764052

Contents

Table of Figures

Table of Tables

(i) Executive Summary

This report, titled “Automated Raw Component Parameter Information Extraction: Algorithms and Methods Comparison” focuses on comparing the efficiency, effectiveness, sustainability and maintainability of the two methods developed for automating the process of parameter information extraction of raw components used by the “Component and Technology” department of Tata Consultancy Service’s San Jose based network and communication based customer. The aim of the automation is the reduction of time spent by skilled engineers in performing mundane tasks, and allowing them to focus at more important tasks at hand.

Research methodology primarily involves the development, and measurement of efficiency and effectiveness of two automation methods, namely “Data sheet extraction”, and “Web Scrapping of e-commerce sites”. The quantitative results are obtained by running the algorithms with test samples and observing the success rate for different factors of the two algorithms. The qualitative comparisons of the two methods allows for a descriptive analysis of the implicit and non-functional factors.

The quantitative analysis concludes that the efficiency of the “Data sheet extraction” method is comparatively higher than the Web-Scrapping algorithm, by 15%. The parameter extraction effectiveness is also higher for “Data sheet extraction” method by 11%. Overall demonstrating that the “Data sheet extraction” method has a higher adaptability for the automation process. The qualitative analysis shows that “Data sheet extraction” method is high in maintenance, and has a low sustainability over the Web Scrapping algorithm. This is due to the non-standard format of the data sheets used across the industry. The report recommends the use of Natural Language Processing for improving the “Data sheet extraction” method, along with other improvements for the Web-Scrapping algorithm.

1. Introduction

One of the significant customers of Tata Consultancy Services (TCS) is a San Jose based multinational Networks and Communications company which develops, manufactures and sells networking hardware, software, telecommunications equipment and other high-technology services and products. (In accordance with the confidentiality agreement with TCS the name of the company has been concealed)

The "Component and Technology" department of the mentioned company is responsible for identifying, evaluating and approving new and versioned up components of Suppliers across the globe. Only the approved hardware component raw materials can be part of the BoM (bill of materials) to manufacture hardware products across manufacturing sites. With growing demand for component raw materials, the number of components to be evaluated is surging over a period of time. Having said that, one of the vital processes involved is to manually extract and consolidate the component parameter, and data from component data specification of respective raw component manufacturers. These raw components are accompanied by their respective data sheets (in PDF format) which include all necessary information and parameters required by the engineers to further approve the raw components. These data sheets are the industry standard and are the most reliable way of obtaining the right component parameters. The lack of an industry standard format of providing such information compels the department to manually extract the relevant information by individual employees going through the entirety of the PDFs.

To reduce the time and money spent on high skilled engineers in performing such mundane and time intensive tasks, the whole process of parameter data extraction and consolidation is required to be automated through multi-channel search to ensure data availability and integrity, which completely eliminates the manual efforts, so that the component engineers can focus more on the technology intensive tasks.

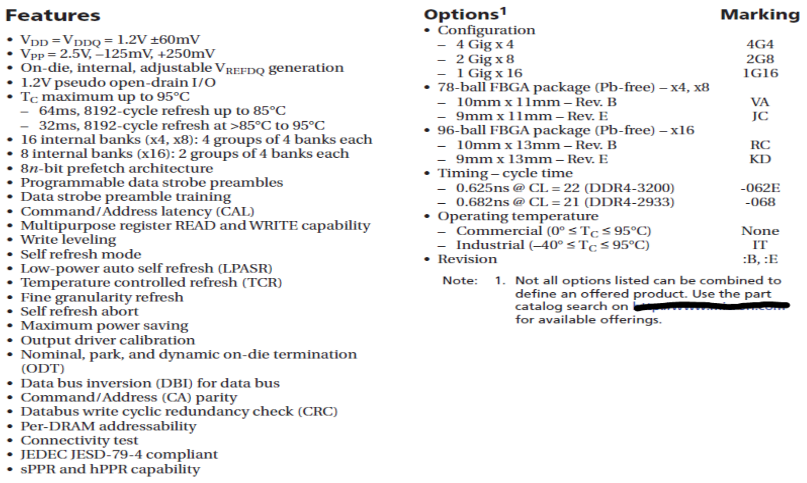
To achieve this goal two primary methods namely “Data sheet extraction”, and “Web Scrapping of e-commerce sites” were investigated and developed. The results of the two methods in extracting the component parameters had various degrees of challenges and variances in efficiencies for different measures. The main aim of this report is to quantitatively and qualitatively examine the differences in efficiencies and overall usefulness of the two methods, and to ultimately determine the better of the two approaches. Recommendations are then made for further improvement of the algorithms, and new methods are also suggested based on deficiencies observed in these methods.

2. Analysis: Data sheet extraction:

This section of the report explores and examines the method developed to automate the process of parameter extraction from data sheets. Various algorithms are explored to improve the efficiency and effectiveness of the parameter extraction

2.1 Extracting Component parameters from Plain text:

As a sample, handpicked data sheets from “Manufacturer A” were used (Name of manufacturer concealed for confidentiality). Format chosen for the data sheets was Portable Data Format due to its wide availability and standard market use. On examination, it is evident that the components parameters that are to be extracted are present in form of plain text in the summary page of the data sheet. For example, as shown in the sample below:



Features

- $V_{DD} = V_{DDQ}$ = 1.2V ±60mV
- V_{PP} = 2.5V, –125mV, +250mV
- On-die, internal, adjustable V_{REFDQ} generation
- 1.2V pseudo open-drain I/O
- T_C maximum up to 95°C
  - 64ms, 8192-cycle refresh up to 85°C
  - 32ms, 8192-cycle refresh at >85°C to 95°C
- 16 internal banks (x4, x8): 4 groups of 4 banks each
- 8 internal banks (x16): 2 groups of 4 banks each
- 8*n*-bit prefetch architecture
- Programmable data strobe preambles
- Data strobe preamble training
- Command/Address latency (CAL)
- Multipurpose register READ and WRITE capability
- Write leveling
- Self refresh mode
- Low-power auto self refresh (LPASR)
- Temperature controlled refresh (TCR)
- Fine granularity refresh
- Self refresh abort
- Maximum power saving
- Output driver calibration
- Nominal, park, and dynamic on-die termination (ODT)
- Data bus inversion (DBI) for data bus
- Command/Address (CA) parity
- Databus write cyclic redundancy check (CRC)
- Per-DRAM addressability
- Connectivity test
- JEDEC JESD-79-4 compliant
- sPPR and hPPR capability

Options[1]	**Marking**
• Configuration	
– 4 Gig x 4	4G4
– 2 Gig x 8	2G8
– 1 Gig x 16	1G16
• 78-ball FBGA package (Pb-free) – x4, x8	
– 10mm x 11mm – Rev. B	VA
– 9mm x 11mm – Rev. E	JC
• 96-ball FBGA package (Pb-free) – x16	
– 10mm x 13mm – Rev. B	RC
– 9mm x 13mm – Rev. E	KD
• Timing – cycle time	
– 0.625ns @ CL = 22 (DDR4-3200)	-062E
– 0.682ns @ CL = 21 (DDR4-2933)	-068
• Operating temperature	
– Commercial (0° ≤ T_C ≤ 95°C)	None
– Industrial (–40° ≤ T_C ≤ 95°C)	IT
• Revision	:B, :E

Note: 1. Not all options listed can be combined to define an offered product. Use the part catalog search on [redacted] for available offerings.

Figure 1: Data Sheet extraction– Plain text

As is observed from the sample taken, a large chunk of parameters can be identified with the help of special text patterns which may be in form of their units, symbol, or descriptions. These unique identifiers provide the flexibility to use standard "Keyword detection" methods for identifying the required component parameters directly from the paragraph.

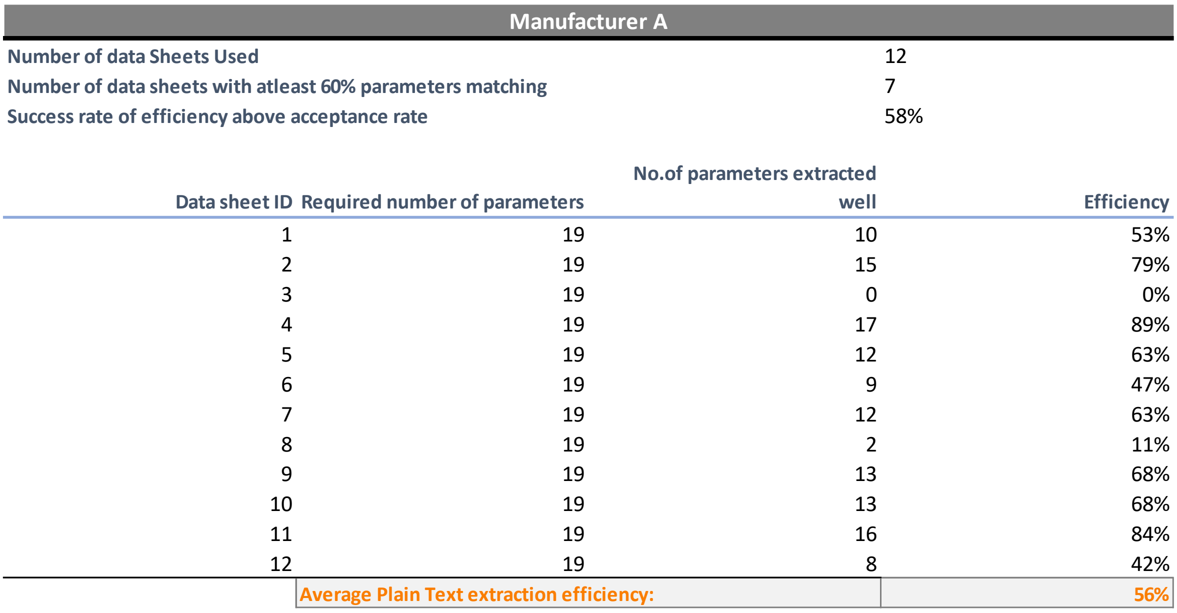
Methods for the text extraction, of plain text with specific patterns, were inspired from the works presented by Leeuwen et al., (1990). The works presented under the sections "Algorithms for finding patterns in strings" provide a multitude of techniques and algorithms for text extraction. Of the works presented, "Regular Expression" ("Regex" here on now) can be considered the most suitable method for this case. Text patterns that follow a user-defined series of characters can be detected using Regex. This allows us to create specified patterns for data that follows specific notations (López & Romero, 2014), as in our case, text matching certain units, symbols, or descriptions.

Programming functions from the python libraries "PyPDF2" and "PDFPlumber" were used for parsing the PDF data sheets, for each set of characters that were separated by spaces. The extracted character sets were iterated and matched with the regex text pattern, using python library "re", until a set of characters that satisfied the Regex were identified. These set of characters were returned as the result of the component parameters, and were exported to a Microsoft Excel sheet using functions from the python Library “xlswriter”.

The above steps were followed for each component parameter, across every page of the supplied data sheet. Each component parameter was matched and extracted using independent and unique Regex pattern strings.

The Graph and Tables below show the results obtained for various data sheets and companies. The standard number of parameters to be extracted were 19, since data sheets for similar raw components were taken.

The extraction of parameters for any PDF was deemed to be successful if 60% or more of the parameters were identified and extracted by the unique regex(s) used in the algorithm.



Manufacturer A	
Number of data Sheets Used	12
Number of data sheets with atleast 60% parameters matching	7
Success rate of efficiency above acceptance rate	58%

Data sheet ID	Required number of parameters	No.of parameters extracted well	Efficiency
1	19	10	53%
2	19	15	79%
3	19	0	0%
4	19	17	89%
5	19	12	63%
6	19	9	47%
7	19	12	63%
8	19	2	11%
9	19	13	68%
10	19	13	68%
11	19	16	84%
12	19	8	42%
	Average Plain Text extraction efficiency:		56%

Table 1: Data sheet extraction- Plain text for manufacturer A

As observed, for Manufacturer A we could sample 12 data sheets, with a 58% success rate. The average efficiency of the extraction mechanism was 55.7%, i.e., on average 55.7% of the required parameters were obtained from the data sheets. The probable cause behind the low success rate and low efficiency was determined to be due to inconsistencies in the format of the data presented by the manufacturer. Other discrepancies observed were the un-systematic use of abbreviations, and, the use of vector diagrams by Manufacturer A for the component parameters, which went undetected in the Plain text-regex method.

Manufacturer B	
Number of data Sheets Used	8
Number of data sheets with atleast 60% parameters matching	3
Success rate of efficiency above acceptance rate	38%

Data sheet ID	Required number of parameters	No.of parameters extracted well	Efficiency
1	19	12	63%
2	19	5	26%
3	19	13	68%
4	19	9	47%
5	19	10	53%
6	19	6	32%
7	19	12	63%
8	19	4	21%
	Average Plain Text extraction efficiency:		47%

Table 2: Data sheet extraction- Plain text for manufacturer B

For Manufacturer B, we could sample 8 data sheets, with a 37.5% success rate. The average effectiveness of the extraction mechanism was 46.7%, i.e., on average 46.7% of the required parameters were obtained from the PDF.

Manufacturer C	
Number of data Sheets Used	9
Number of data sheets with atleast 60% parameters matching	2
Success rate of efficiency above acceptance rate	22%

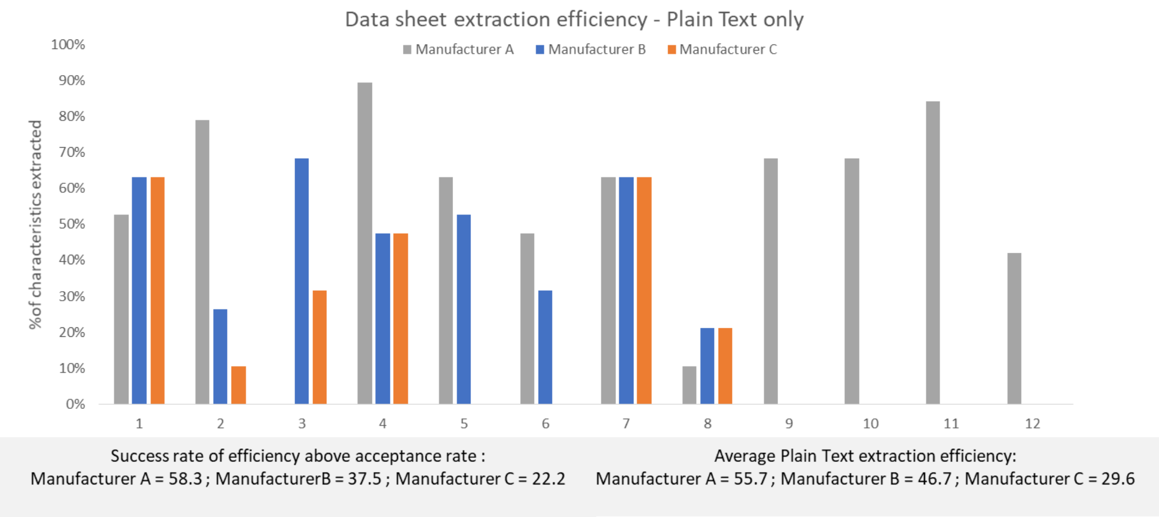
Data sheet ID	Required number of parameters	No.of parameters extracted well	Efficiency
1	19	12	63%
2	19	2	11%
3	19	6	32%
4	19	9	47%
5	19	0	0%
6	19	0	0%
7	19	12	63%
8	19	4	21%
	Average Plain Text extraction efficiency:		30%

Table 3: Data sheet extraction- Plain text for manufacturer C

For Manufacturer C, we could sample 9 data sheets, with a 22.2% success rate. The average effectiveness of the extraction mechanism was 29.6%, i.e., on average 29.6% of the required parameters were obtained from the PDF.

The primary reason behind the low success rate and efficiency for Manufacturer B and C was observed to be that most of the parameters were not described as plain text in the component descriptions, but were primarily presented in the form of tables, which went undetected by the text extraction functions of the python libraries.

The above results are condensed into the following graph for convenience:



Success rate of efficiency above acceptance rate :
Manufacturer A = 58.3 ; ManufacturerB = 37.5 ; Manufacturer C = 22.2

Average Plain Text extraction efficiency:
Manufacturer A = 55.7 ; Manufacturer B = 46.7 ; Manufacturer C = 29.6

Figure 2: Graph - Data sheet extraction summary - Plain text

2.2 Extracting Component details from diagram:

The overall low efficiency in extracting component parameters from Manufacturer A could be associated to the repeated use of vector diagrams for providing the information. The extraction of text from such diagrams led to multiple issues such as lack of spaces between symbols and characters, scrambling of characters, or indistinguishable start and end for required character strings. If text from the diagram was extracted well, further challenge remained in associating the correct symbols to the correct descriptions in the diagram as shown below in the sample.

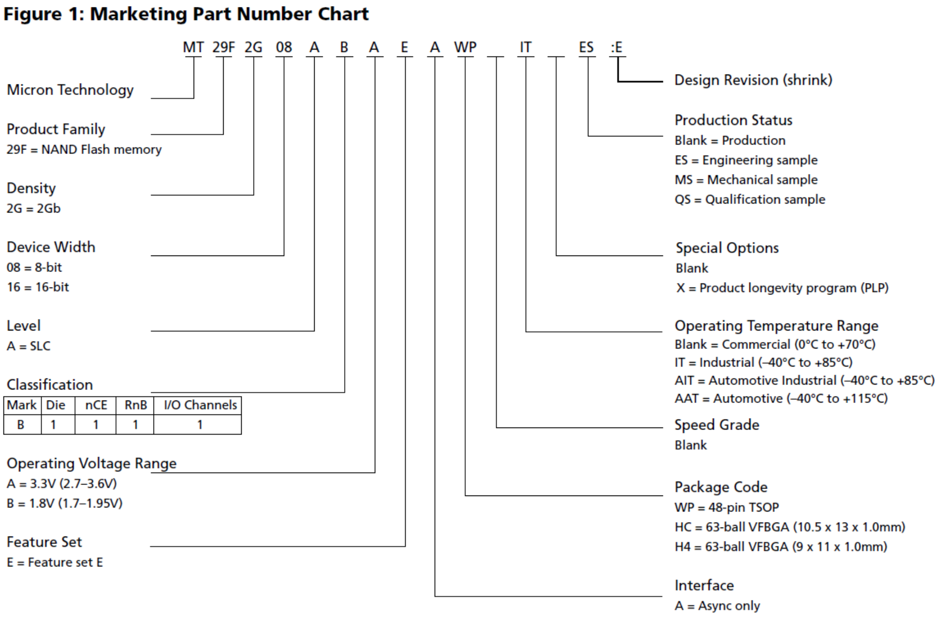


Figure 3: Data Sheet– Parameter Diagram

To overcome the issues of poor text extraction and description matching, a custom method was developed. Under this method, the PDF was directly converted into a Microsoft Excel file using multiple methods with variable degrees of success. The conversion of the PDF text to Excel allowed for the data available in the diagram to be extracted and separated in independent cells of the excel worksheet. The PDF page on which the diagram was present was mostly consistent, hence certain cells could be predicted in which there was the highest probability of finding the parameters to be extracted. The following methods were used for PDF to Excel conversion:

- Tabula: Tabula is python library. The conversion quality was overall poor. Most data was left unconverted in the resulting excel file
- Pandas: Pandas is a standard python library used for data analysis. The quality of conversion had room for improvement. Many characters were scrambled and there was inconsistency between the cells in which the output was obtained.
- Adobe pdf to excel converter API: The API provided the best quality of PDF to Excel conversion. Limitations included data being extracted in inconsistent cell IDs, the

service was also subscription based, which involved varying levels of financial implication depending on licenses cost.

The Parameter Diagram Ribbon consisting of parameter keys needed to be matched with the correct description of the key (refer to Figure 3) for obtaining relevant information.

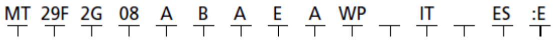


Figure 4: Data Sheet– Parameter Diagram Ribbon

The keys in the ribbon were iterated from left to right. The left most key was mapped to the top most description of the left column, followed by the second key from left being mapped to the second description in the left column and so on. This was iterated until the last row of description was reached. A similar process was followed for the description in the right column of the diagram with the key mapping going from right to left. Once each key was associated with the respective description, the appropriate regex pattern was used to extract the parameters name and the key label was used to extract the value for the parameter.

The following improved results were obtained with the above implementations:

Manufacturer A			
Number of data Sheets Used		12	
Number of data sheets with atleast 60% parameters matching		10	
Success rate of efficiency above acceptance rate		83%	
Data sheet ID	Required number of parameters	No.of parameters extracted well	Efficiency
1	19	12	63%
2	19	15	79%
3	19	10	53%
4	19	17	89%
5	19	15	79%
6	19	11	58%
7	19	12	63%
8	19	14	74%
9	19	14	74%
10	19	13	68%
11	19	16	84%
12	19	11	58%
	Average Plain Text + Diagram Text extraction efficiency:		70%

Table 4: Data sheet extraction- Plain text and parameter diagram for manufacturer A

The data extraction from diagram improved the success rate from the previous 58% to the new 83% success. The overall efficiency improved from the previous 55.7% to 70.2% on average. i.e., on average 70.2% of the parameters were extracted from each PDF.

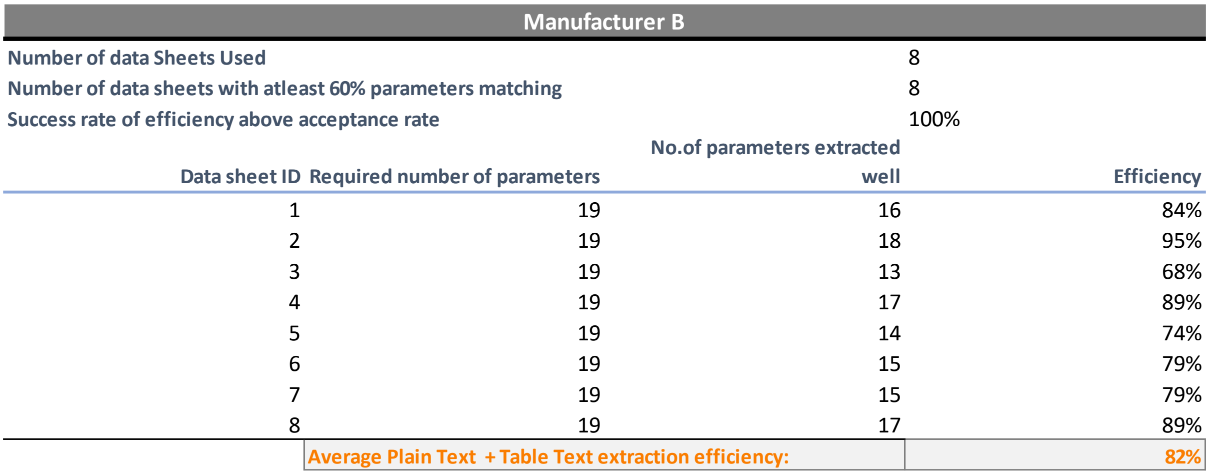
Despite the improved efficiency and effectiveness, the method has various limitations. The diagram data extraction is limited to the data sheets by Manufacturer A. Most other manufacturers do not use similar diagrams or diagrams at all. Within the documents of manufacturer A, there are multiple inconsistencies among data sheets which requires constant error-handling and code changes. This reduces the versatility and reliability of the program. From the extracted diagram data, many parameters were absent in the final result for various reasons despite the data being available in the PDF.

2.3 Extracting Component details from description tables:

On observation it is made clear that manufacturers B and C primarily provide their component descriptions in tabular forms. Text and information in these tables goes unrecognized by the text extraction libraries. Different python libraries were used such that the tables present in the PDF documents could be extracted with ease. The following libraries were used:

- Tabula in support with Pandas: Tabula, a python library, was used to extract the information from the PDF. The extracted information was converted into a tabular form using the Pandas library. Overall efficiency of the method was low as many tables were not extracted. Other issues included row mismatches, and lost information.
- Camelot: Camelot is a python library dedicated to extract tables from PDFs. Camelot provided high efficiency and reliable results in terms of the tables extracted.

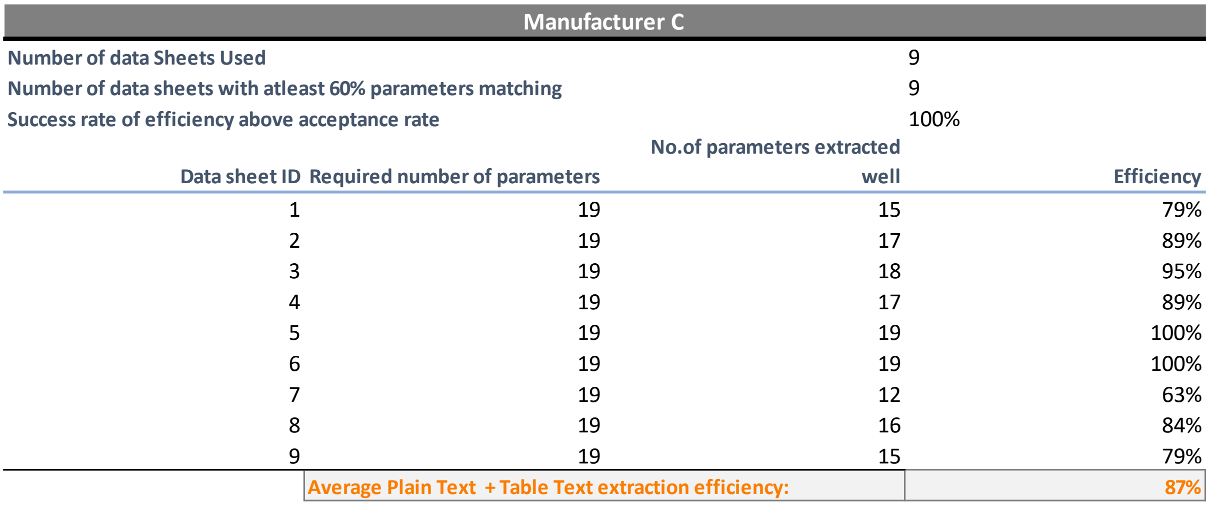
Once the tables were obtained using Camelot, Pandas was used to parse through the extracted tables cell by cell. The rows of the first column of every table were iterated and matched with the regex patterns of the specific parameters that were required. Once found in the first column, the respective values from the remaining columns of the same row were exported to excel. The following results were obtained from the table extraction method:



Manufacturer B			
Number of data Sheets Used		8	
Number of data sheets with atleast 60% parameters matching		8	
Success rate of efficiency above acceptance rate		100%	
Data sheet ID	Required number of parameters	No.of parameters extracted well	Efficiency
1	19	16	84%
2	19	18	95%
3	19	13	68%
4	19	17	89%
5	19	14	74%
6	19	15	79%
7	19	15	79%
8	19	17	89%
	Average Plain Text + Table Text extraction efficiency:		82%

Table 5: Data sheet extraction- Plain text and Table for manufacturer B

The success rate rose from 37.5% to 100% for manufacturer B. The overall efficiency rose from 46.7% to 82.2%, i.e., on average 82.2% of the parameters were obtained from the PDFs.

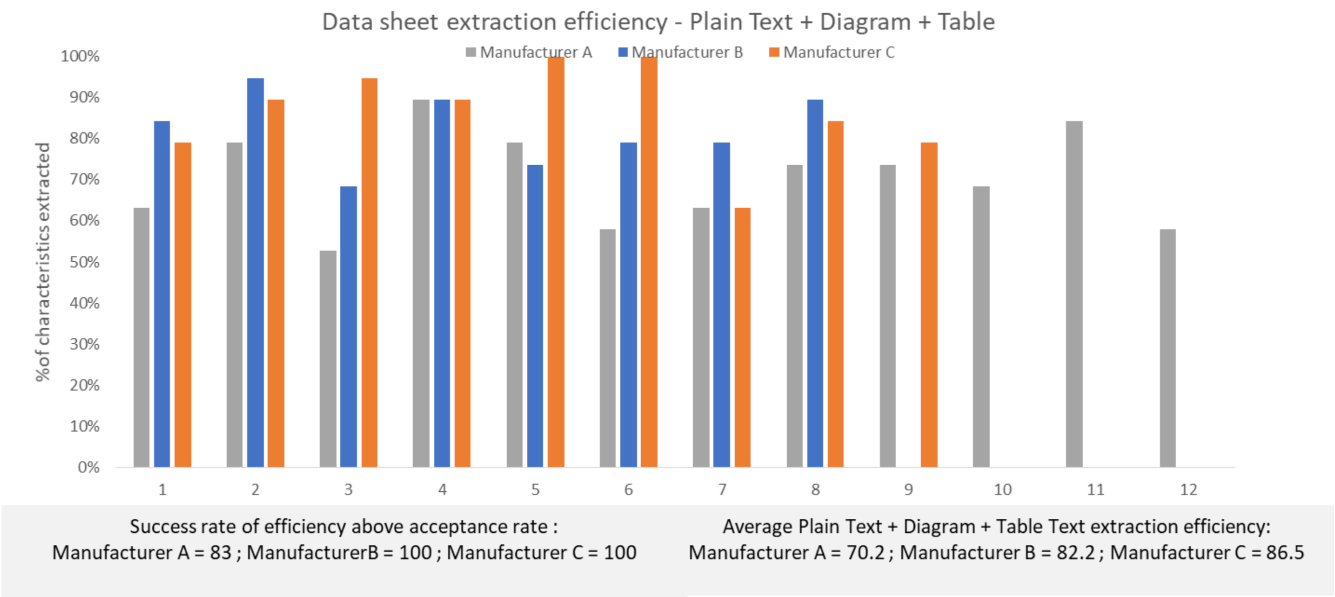


Manufacturer C			
Number of data Sheets Used		9	
Number of data sheets with atleast 60% parameters matching		9	
Success rate of efficiency above acceptance rate		100%	
Data sheet ID	Required number of parameters	No.of parameters extracted well	Efficiency
1	19	15	79%
2	19	17	89%
3	19	18	95%
4	19	17	89%
5	19	19	100%
6	19	19	100%
7	19	12	63%
8	19	16	84%
9	19	15	79%
	Average Plain Text + Table Text extraction efficiency:		87%

Table 6: Data sheet extraction- Plain text and Table for manufacturer C

The success rate rose from 22.2% to 100% for manufacturer C. The overall efficiency rose from 29.6% to 86.5%, i.e., on average 86.5% of the parameters were obtained from the PDFs. This high increase in success rate and efficiency is due to the high availability of information in form of tables for manufacturer C data sheets.

The above results are condensed into the following graph for convenience:



Success rate of efficiency above acceptance rate :
Manufacturer A = 83 ; ManufacturerB = 100 ; Manufacturer C = 100

Average Plain Text + Diagram + Table Text extraction efficiency:
Manufacturer A = 70.2 ; Manufacturer B = 82.2 ; Manufacturer C = 86.5

Figure 5: Graph- Improved Data sheet extraction summary - Plain text, Parameter Diagram, and Table

The methods were overall successful. The primary issues remaining include lack of consistency in tables and diagram formats among manufacturers, due to lack of industry standards. Different formats across the industry make it necessary to customize the regex, table iterations and overall algorithm for each manufacturer. This prevents versatility in the methods.

3. Analysis: Web scrapping of e-commerce sites

Data sheets are the most suitable form of retrieving component descriptions. However, the lack of an industry standard format makes it difficult to automate the task of retrieving such information from the data sheets. Despite the attempts made during the work term towards automation of the task, the lack of industry standard formats makes it necessary to create unique methods of extraction for every type of data sheet sent by a manufacturer. This variance across the data sheets and high number of manufacturers with their unique formats demands an extremely large number of unique methods to be created. While this process may be do-able, it is extremely time and resource consuming, creating an imbalance in the overall benefit of such an automation. It hence becomes important to create a process which will remain standard across various manufacturers and components.

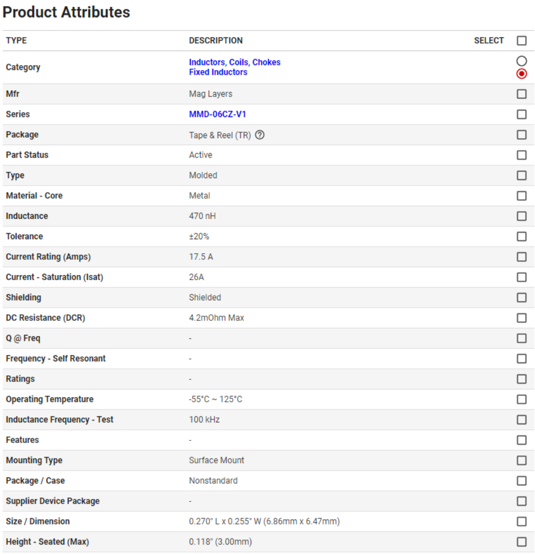
Upon exploring, ecommerce websites that supplied electronic engineering components were identified to provide component descriptions for most of the components required, across all manufacturers. Since the ecommerce suppliers had a standard format to provide component description as per the site specification, the issue of varying formats of data provision could be eliminated by directly reading component parameters off such websites.

Using methods of Web Scrapping, automation test scripts can be written which allow extraction and gathering of information without the intervention of manual efforts or the use of APIs (Mitchell, 2018). Such test systems can allow us to access the websites of the mentioned ecommerce suppliers, and directly utilise the information on their web-pages related to the selected component parameters.

Two ecommerce suppliers, “digikey.com” and “mouser.com” were identified as reliable and relevant source of component descriptions. To access the information of the required component, a unique identifier known as “Manufacturer Part Number” (MPN) needed to be used to reach to the specific component’s URL.

3.1 BeautifulSoup4

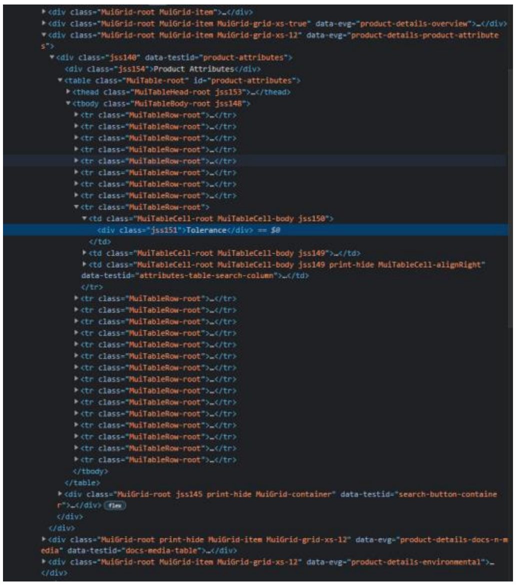
The basic methods as described by Mitchell (2018) for web scrapping, allow the program to access the HTML code of the specified URL and parse the HTML page to obtain the required text. The two mentioned websites displayed information in HTML table format. To detect the tables, the HTML table tag IDs were identified using the python library BeautifulSoup4 (BS4). BS4 was chosen due to it’s almost industry standard use for web scrapping. The use of BS4 allowed for parsing through the table to extract the component parameters.



Product Attributes

TYPE	DESCRIPTION	SELECT □
Category	Inductors, Coils, Chokes Fixed Inductors	○ ●
Mfr	Mag Layers	□
Series	MMD-06CZ-V1	□
Package	Tape & Reel (TR) ⓘ	□
Part Status	Active	□
Type	Molded	□
Material - Core	Metal	□
Inductance	470 nH	□
Tolerance	±20%	□
Current Rating (Amps)	17.5 A	□
Current - Saturation (Isat)	26A	□
Shielding	Shielded	□
DC Resistance (DCR)	4.2mOhm Max	□
Q @ Freq	-	□
Frequency - Self Resonant	-	□
Ratings	-	□
Operating Temperature	-55°C ~ 125°C	□
Inductance Frequency - Test	100 kHz	□
Features	-	□
Mounting Type	Surface Mount	□
Package / Case	Nonstandard	□
Supplier Device Package	-	□
Size / Dimension	0.270" L x 0.255" W (6.86mm x 6.47mm)	□
Height - Seated (Max)	0.118" (3.00mm)	□

Figure 6: Web Scrapping HTML source



```
▶<div class="MuiGrid-root MuiGrid-item">…</div>
▶<div class="MuiGrid-root MuiGrid-item MuiGrid-grid-xs-true" data-evg="product-details-overview">…</div>
▼<div class="MuiGrid-root MuiGrid-item MuiGrid-grid-xs-12" data-evg="product-details-product-attribute
s">
  ▼<div class="jss140" data-testid="product-attributes">
     <div class="jss154">Product Attributes</div>
    ▼<table class="MuiTable-root" id="product-attributes">
      ▶<thead class="MuiTableHead-root jss153">…</thead>
      ▼<tbody class="MuiTableBody-root jss148">
        ▶<tr class="MuiTableRow-root">…</tr>
        ▶<tr class="MuiTableRow-root">…</tr>
        ▶<tr class="MuiTableRow-root">…</tr>
        ▶<tr class="MuiTableRow-root">…</tr>
        ▶<tr class="MuiTableRow-root">…</tr>
        ▶<tr class="MuiTableRow-root">…</tr>
        ▶<tr class="MuiTableRow-root">…</tr>
        ▶<tr class="MuiTableRow-root">…</tr>
        ▼<tr class="MuiTableRow-root">
          ▼<td class="MuiTableCell-root MuiTableCell-body jss150">
             <div class="jss151">Tolerance</div> == $0
           </td>
          ▶<td class="MuiTableCell-root MuiTableCell-body jss149">…</td>
          ▶<td class="MuiTableCell-root MuiTableCell-body jss149 print-hide MuiTableCell-alignRight"
         data-testid="attributes-table-search-column">…</td>
         </tr>
        ▶<tr class="MuiTableRow-root">…</tr>
        ▶<tr class="MuiTableRow-root">…</tr>
        ▶<tr class="MuiTableRow-root">…</tr>
        ▶<tr class="MuiTableRow-root">…</tr>
        ▶<tr class="MuiTableRow-root">…</tr>
        ▶<tr class="MuiTableRow-root">…</tr>
        ▶<tr class="MuiTableRow-root">…</tr>
        ▶<tr class="MuiTableRow-root">…</tr>
        ▶<tr class="MuiTableRow-root">…</tr>
        ▶<tr class="MuiTableRow-root">…</tr>
        ▶<tr class="MuiTableRow-root">…</tr>
        ▶<tr class="MuiTableRow-root">…</tr>
        ▶<tr class="MuiTableRow-root">…</tr>
        ▶<tr class="MuiTableRow-root">…</tr>
        ▶<tr class="MuiTableRow-root">…</tr>
       </tbody>
     </table>
    ▶<div class="MuiGrid-root jss145 print-hide MuiGrid-container" data-testid="search-button-containe
    r">…</div> flex
   </div>
 </div>
▶<div class="MuiGrid-root print-hide MuiGrid-item MuiGrid-grid-xs-12" data-evg="product-details-docs-n-m
edia" data-testid="docs-media-table">…</div>
▶<div class="MuiGrid-root MuiGrid-item MuiGrid-grid-xs-12" data-evg="product-details-environmental">…
</div>
```

Figure 7: Web Scrapping Parameter table

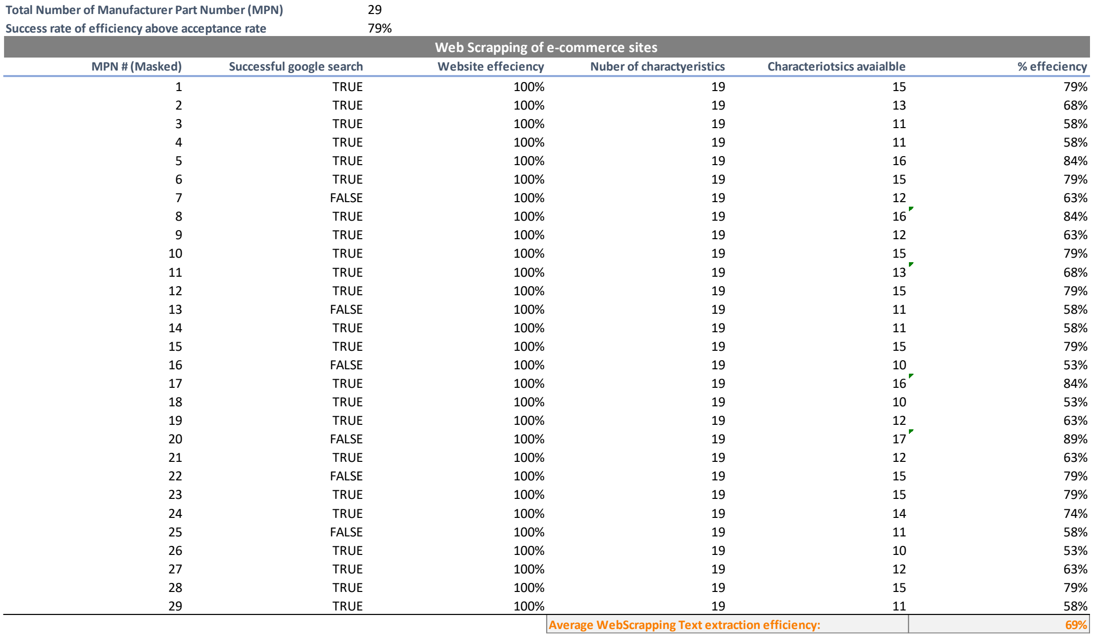
Limitations in this method included inconsistency in recognizing table tags and text extraction from the table cells during iterations. Apart from low efficiency, BS4 required the exact URL to be specified for scrapping. The requirement of manually searching the required website URL defeats the purpose of the automation.

3.2 Selenium and Google

To resolve the limitations of manually entering the component URL, python's "Google" library was used to perform automated searches for the list of MPNs provided as user Input. The Google library was programmed to return up to 5 URLs that had any mention of the required web sites, namely "DigiKey" and "Mouser". If in the top 5 results these websites did not have the required MPNs listed, the function returned a null value. However, in most of the cases if the web site was found, preference was given to DigiKey followed by Mouser.

To overcome the issues of low efficiency of BS4, methods as described by Wu et al. (2020) were used for creating a test automation for the Google chrome browser. The selenium web driver was used to control the chrome browser and open the respective URLs as provided by the Google library search results. Upon accessing the webpage, Selenium and BS4 were used to locate the elements in the webpage using a unique identifier for any web element known as "Xpath", In contrast to the use of HTML tags which provided low efficiency and inconsistent results. Using the XPath(s), the exact table and cell ID could be retrieved and the component parameters present in them were accessed by iterating through the table cells. These values were then exported back to an excel file using python's "xlswriter" library.

The MPNs of the components from Manufacturer A, B, C were again used and put through the web scrapping method to retrieve the component parameters with the following results:



Total Number of Manufacturer Part Number (MPN)	29
Success rate of efficiency above acceptance rate	79%

Web Scrapping of e-commerce sites					
MPN # (Masked)	Successful google search	Website effeciency	Nuber of charactyeristics	Characteriotsics avaialble	% effeciency
1	TRUE	100%	19	15	79%
2	TRUE	100%	19	13	68%
3	TRUE	100%	19	11	58%
4	TRUE	100%	19	11	58%
5	TRUE	100%	19	16	84%
6	TRUE	100%	19	15	79%
7	FALSE	100%	19	12	63%
8	TRUE	100%	19	16	84%
9	TRUE	100%	19	12	63%
10	TRUE	100%	19	15	79%
11	TRUE	100%	19	13	68%
12	TRUE	100%	19	15	79%
13	FALSE	100%	19	11	58%
14	TRUE	100%	19	11	58%
15	TRUE	100%	19	15	79%
16	FALSE	100%	19	10	53%
17	TRUE	100%	19	16	84%
18	TRUE	100%	19	10	53%
19	TRUE	100%	19	12	63%
20	FALSE	100%	19	17	89%
21	TRUE	100%	19	12	63%
22	FALSE	100%	19	15	79%
23	TRUE	100%	19	15	79%
24	TRUE	100%	19	14	74%
25	FALSE	100%	19	11	58%
26	TRUE	100%	19	10	53%
27	TRUE	100%	19	12	63%
28	TRUE	100%	19	15	79%
29	TRUE	100%	19	11	58%
			Average WebScrapping Text extraction efficiency:		69%

Table 7: Web Scrapping Extraction: Selenium results

Of the total 29 MPNs searched, 23 were available on either mentioned websites. Giving the availability efficiency of 79.3%. For all the components, 100% of parameters that were present on the website were retrieved, showing that the method worked well in terms of the technical process. However, In terms of availability of required parameters on the websites, only 69.2%

of the required parameters were available. This implies that on average around 30% of the required parameters are not available on the webpages that were being scraped.

4. Analytic discussion and Conclusion:

The report summarizes the results obtained for the two different methods explored for the automation of component parameter extraction required by TCS customer's "Component and Technology" department for the purpose of using these approved components in their final BoM (Bill of Materials). The two methods "Data Sheet extraction", and "Web Scrapping of e-commerce sites" were investigated and developed during the term.

The two methods explored during the work term are distinctively independent in their methodologies, resource requirement, time, effort as well as the quality of output received. This section of the report focuses primarily on the comparison of the quality of results obtained, the sustenance, maintenance and modification efforts potentially required in the future, and the overall better of the two methods.

The following table summarizes the output efficiency:

Data sheet Extraction (Plain text + Diagram + Table)				
	Manufacturer A	Manufacturer B	Manufacturer C	Average efficiency
Effeciency above	83%	100%	100%	94%
Average effectiveness in parameter extraction	70%	82%	87%	80%

Web Scrapping of e-commerce sites	
Effeciency of MPN availbility	79%
Effeciency of extraction	100%
Effeciency of parameter availability	69%

Table 8: Final results summary

For the data sheet PDF extraction algorithm, a component's data sheet was considered to be successfully extracted if at least 60% of the parameter values were obtained. A threshold of 60% was maintained as any number of parameters below it would require enough manual effort to extract the rest of the information that the automation may as well be considered void. With this threshold, information for 94.4% of the components was successfully extracted. In contrast, the Web-Scrapping method could only extract information for 79.3% of the components.

For each individual component, the data sheet PDF extraction algorithm gave an efficiency of 79.7%, i.e, on average 79.7% of the parameters for each component were found and extracted by the algorithm. The Web Scrapping algorithm on the other was on average only successful in extracting 69.2% of the parameters for each component.

The quantitative comparison of the methods clearly shows that the data sheet extraction algorithm provides a much better result, both in terms of successful number of components whose information was extracted, as well as the number of parameters extracted for each component.

Comparing the qualitative attributes related to sustainability and maintainability, the data sheet extraction algorithm falters due to the non-standardization of the data sheet format. This creates a constant requirement of creating new algorithms for every change in format that is experienced during extraction i.e., Any change in format by the manufacturer, or new addition of the manufacturers will require the algorithms to be redeveloped, with new creative methods required to cater to each change. This form of automation is difficult to sustain. In contrast, in case of web scraping all manufacturers are expected to provide component parameters in the format as specified by the website. This allows algorithms that do not require constant modification overtime. This increases the sustainability and reduces the maintenance efforts for the algorithm development. However, the process of Web Scrapping is extremely time consuming and has room for performance improvement.

Overall, the Data sheet extraction proves to be a superior method, which requires major improvements in terms of development of an algorithm that can be sustainably used across varying data sheet formats. The use of Web scrapping, despite its rather standard format and marginally acceptable results cannot be used extensively due to the high dependence on external sources for the internally required information.

5. Recommendations:

As is visible from the quantitative results, data sheet PDF extraction algorithm gives us a better efficiency in comparison to web-scraping. Since data sheets are the best and most reliable source of information for the component descriptions, it will be preferable to improve the methodologies and algorithms used for the PDF extraction. The current issue of non-standard formats and text written in natural language can be overcome by the future use of algorithms based on Natural Language Processing (NLP) as expressed by Conlon et al. (2016).

Due to the complexity of the required algorithms as well as the nascent state of the NLP Technology, Web Scraping cannot be ruled out as a viable option. To improve the processing speed of the web scraping algorithms, use of dynamic browser drivers can be employed as mentioned in García et al. (2021). Using other tags apart from the standard XPath to identify web elements can also improve the speed of web scraping (Mitchell, R. E., 2018).

To improve the efficiency of the results obtained from web scraping a wider variety of websites containing component parameters can be used. In such a method it will be important to make sure that no information gets overlapped or duplicated from other websites. Website APIs can also be used as an alternative to web scraping.

However, while furthering the use of web scraping it is important to consider the legal as well as moral consequences of extraction of information from websites without the availability of a valid license to do so. While there may be no current laws against such algorithms, amendments in acts such as the CFAA in the United States can hamper use of such technology (Macapinlac, T., 2019).

(ii) Citations

Conlon, S. J., Hale, J. G., Lukose, S., & Strong, J. (2016). Information Extraction Agents for Service-Oriented Architecture Using Web Service Systems: A Framework. *Journal of Computer Information Systems, 48*, 75–77.

García, B., Munoz-Organero, M., Alario-Hoyos, C., & Kloos, C. D. (2021). Automated driver management for selenium WebDriver. *Empirical Software Engineering*, *26*(5). https://doi.org/10.1007/s10664-021-09975-3

Leeuwen, J. van. (1990). Algorithms for finding patterns in strings. In *Handbook of theoretical computer science* (Vol. A, pp. 282–300). essay, The MIT Press.

López Felix, & Romero, V. (2014). *Mastering Python regular expressions: Leverage regular expressions in Python even for the most complex features*. Packt.

Macapinlac, T. (2019). The legality of web scraping: A proposal. Federal Communications Law Journal, 71(3), 399-0_7,0_8. Retrieved from http://search.proquest.com.proxy.lib.uwaterloo.ca/scholarly-journals/legality-web-scraping-proposal/docview/2326822684/se-2?accountid=14906

Mitchell, R. E. (2018). *Web scraping with python: Collecting data from the modern web*. O'Reilly Media.

Wu, H., Liu, F., Zhao, L., &amp; Shao, Y. (2020). Data Analysis and crawler application implementation based on Python. 2020 International Conference on Computer Network, Electronic and Automation (ICCNEA). https://doi.org/10.1109/iccnea50255.2020.00086