SRM INSTITUTE OF SCIENCE AND TECHNOLOGY
SCHOOL OF COMPUTING
DEPARTMENT OF DATA SCIENCE AND BUSINESS SYSTEMS

18CSC304J COMPILER DESIGN

MINI PROJECT REPORT

EXTRACTING DATA USING REGULAR EXPRESSIONS

Team Members

Name	Registration Number
Prateek Anand	RA1911027010042
Saksham Bansal	RA1911027010054
Snehith Reddy K.	RA1911027010055

CONTENTS

ABSTRACT

We must extract necessary information from reports, articles, papers, websites, etc. These extractions are part of text mining and are essential in converting unstructured data to a structured form which are later used in many applications related to analytics and machine learning.

Such entity extractions use approaches like ‘lookup based’ in which words from documents are searched against a pre-defined data dictionary, ‘rules based’ in which pattern searches are made to find key information, and ‘statistical approach’ in which supervised and unsupervised methods are used to extract the information.

Regular expressions (RedEx) is one of the rules-based pattern search methods. They provide a mechanism to select specific strings from a set of character strings. We have developed a mining tool that uses python scripting combined with regex, which extracts the necessary information of the users mentioned in any textual data.

INTRODUCTION AND MOTIVATION

It is not uncommon for us to need to extract text from any document. It is fairly straightforward for documents that are small in size to extract the data manually. However, for documents with huge amounts of data, it is a very difficult task to extract any useful information required. For this, we had to do something. So in the interest of extracting the data programmatically, we started a brief investigation of the various operations involved in extracting raw text.

There were some challenges involved which included but were not limited to different amounts of address or program information; there was a lot of variation in those lines at the end of addresses, most phone numbers were

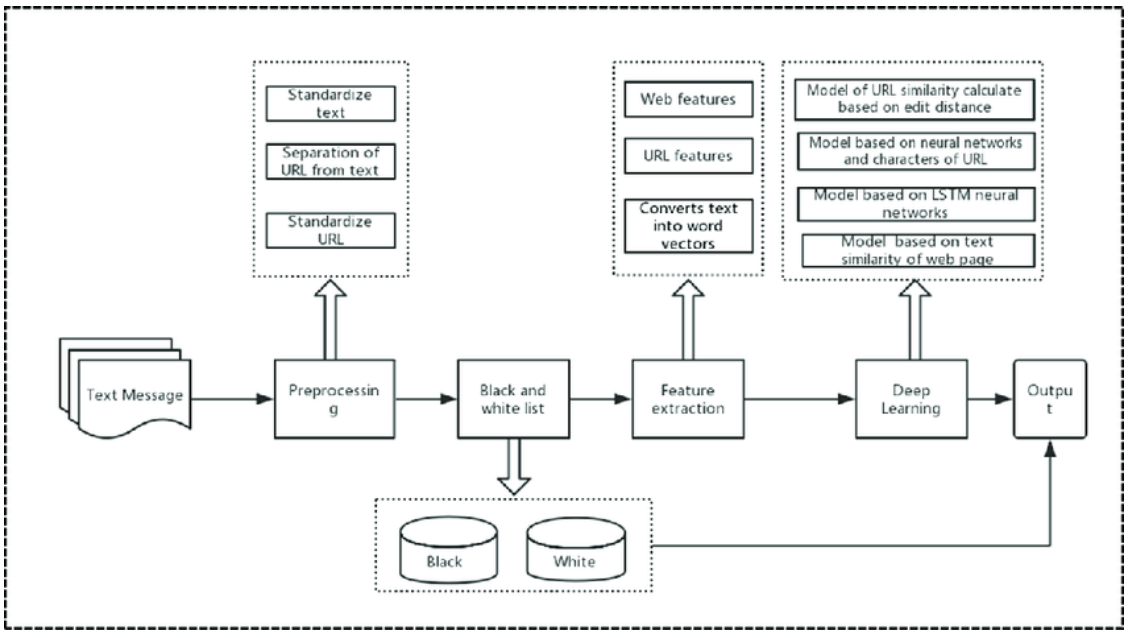
following consistent formatting, but for some, we needed to pull out 1 or 2 additional numbers.

The wealth of data in the repositories calls for an automated technique to retrieve the desired systems from hundreds of thousands of systems. One of the techniques used for data retrieval is Regular expressions (Regex), combined with Python to mine the data and, later, store the data in HTML and CSV documents.

LIMITATION OF EXISTING METHODS

There are many ways to extract information from a text file. But to properly extract information with the pre-existing methods, we need a structured form of data or consistent data. Extracting data with raw text is not ideal as it does not give enough detail or 'signposts' to work with, so formatting patterns are needed in this case. Then came the process, which converts the entire data into a huge text chunk which is nothing but a huge mess.

FLOW DIAGRAM



MODULES

Divided into three sections, reviewing three simple functions to extract useful information from strings with examples:

- **findall()**

  Regex's *findall()* function is extremely useful as it returns a list of strings containing all matches. If the pattern is not found, *re.findall()* returns an empty list.The *findall()* function takes two parameters, the first is the pattern being searched and the second parameter is text we are searching through. This function returns all the non-overlapping matches of the pattern which is in variables, from the second parameter.

```
names = re.findall(r"[A-Za-z]*\s?\-?[A-Za-z]+\,\s?[A-Za-z]+\-?\.?\s?[A-Za-z]*\.?", file_content)
roles = re.findall(r'\b([A-Z]?[a-z]*\s?[A-Z]?[a-z]*\s?[A-Z][a-z]+\s?[A-Z]?[a-z]*)<br\b', file_content)
emails = re.findall(r'\b([A-Za-z0-9]+\.?[A-Za-z0-9]*\.?[A-Za-z0-9]*@[a-z]*\.?utdallas.edu)\s?</a\b', file_content)
contacts = re.findall(r'\-([0-9]{4})', file_content)
```

These are the different regular expressions that we have used in our project :

-> Name

re.findall(r"^[A-Za-z]*\s?\-?[A-Za-z]+\,\s?[A-Za-z]+\-?\.?\s?[A-Za-z]*\.?", file_content)

->Role

re.findall(r'\b([A-Z]?[a-z]*\s?[A-Z]?[a-z]*\s?[A-Z][a-z]+\s?[A-Z]?[a-z]*)<br\b', file_content)

->Email

re.findal l(r'\b([A-Za-z0-9]+\.?[A-Za-z0-9]*\.?[A-Za-z0-9]*@[a-z]*\.?utdallas.edu) \s ?< /a\b', file_content)

->Phone Number

re.findall(r'\-([0-9]{4})', file_content)

- **readLine()**

  readline() reads one entire line from the file. A trailing newline character is kept in the string. If the size argument is present and non-negative, it is a maximum byte count including the trailing newline and an incomplete line may be returned.

```
if input_type=='File Upload':
    file_content = input_file.readline().decode("utf-8")
else:
    print("Error")
```

- **pop()** and **write()**

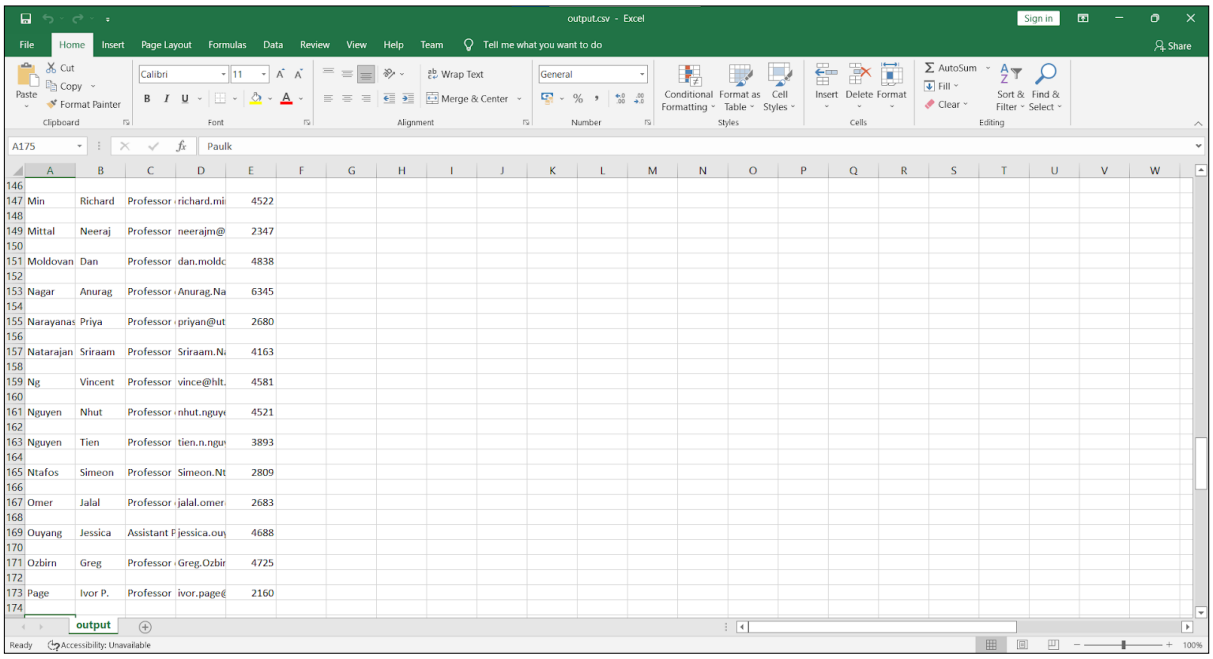
  The pop() method removes the item at the given index from the list and returns the removed item.

  write() writes a string *str* to the file. There is no return value. Due to buffering, the string may not actually show up in the file until the flush() or close() method is called.

```
if names:
    wr(",")
    # print("\n")
    # print("%d" % (i+1), end="\t")
    # print(names.pop(0), end="\t")
    wr(names.pop(0))
    i = i + 1
if roles:
    # print(roles.pop(0), end="\t")
    wrt(",")
    wrt(roles.pop(0))
if emails:
    wrt(",")
    # print(emails.pop(0), end="\t")
    wrt(emails.pop(0))
if contacts:
    # print(contacts.pop(0), end="\t")
    wrt(",")
    wrt(contacts.pop(0))
if file_content == "</html>":
    break
```

SCREENSHOTS

- Excel Sheet

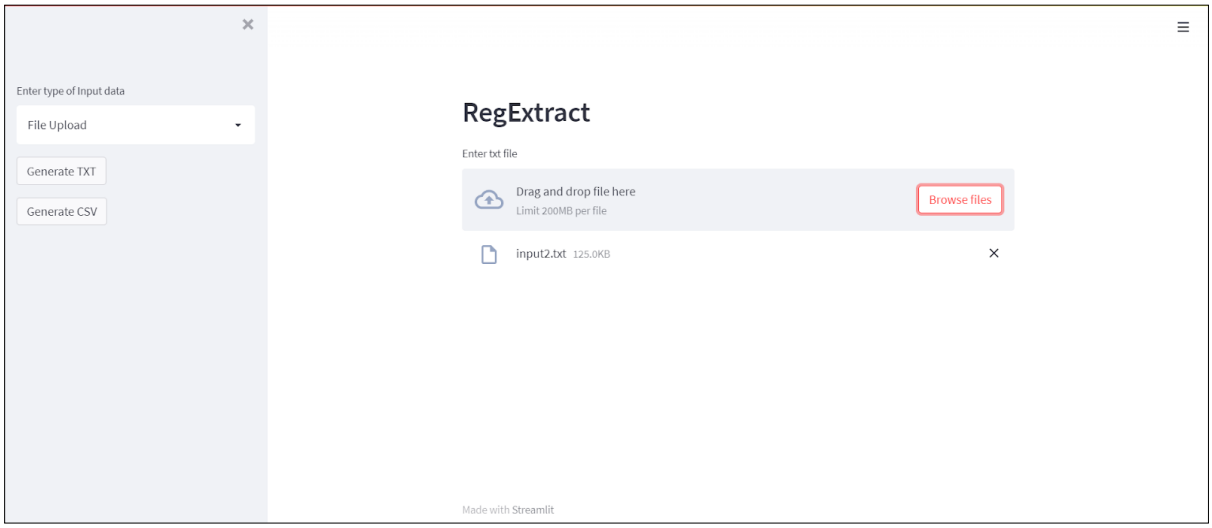


147	Min	Richard	Professor	richard.mi	4522
148					
149	Mittal	Neeraj	Professor	neerajm@	2347
150					
151	Moldovan	Dan	Professor	dan.moldc	4838
152					
153	Nagar	Anurag	Professor	Anurag.Na	6345
154					
155	Narayanas	Priya	Professor	priyan@ut	2680
156					
157	Natarajan	Sriraam	Professor	Sriraam.Na	4163
158					
159	Ng	Vincent	Professor	vince@hlt.	4581
160					
161	Nguyen	Nhut	Professor	nhut.nguye	4521
162					
163	Nguyen	Tien	Professor	tien.n.ngu	3893
164					
165	Ntafos	Simeon	Professor	Simeon.Nt	2809
166					
167	Omer	Jalal	Professor	jalal.omer	2683
168					
169	Ouyang	Jessica	Assistant P	jessica.ou	4688
170					
171	Ozbirn	Greg	Professor	Greg.Ozbir	4725
172					
173	Page	Ivor P.	Professor	ivor.page@	2160

Expanded view of the CSV file

\

- Web App


Enter type of Input data
File Upload
Generate TXT
Generate CSV
RegExtract
Enter txt file
Drag and drop file here
Limit 200MB per file
Browse files
input2.txt 125.0KB
Made with Streamlit


RegExtract
Enter txt file
Drag and drop file here
Limit 200MB per file
Browse files
input2.txt 125.0KB
Text file is successfully converted to CSV format

CONCLUSION

RegEx is a versatile, portable, and powerful way of extracting key information from textual data. The advantage of using this technique is that it can save a significant amount of time on the data collection process. Future work includes searching multiple repositories to improve the data retrieval process and enhance search efficiency and effectiveness. Mastery over it can help automate many mundane tasks. Although it can sometimes get complicated and hard to develop-debug at times, owing to its immense capabilities, it has become a must-have weapon in every programmer's armor, especially for text analytics data scientists.

REFERENCES

- https://www.tutorialspoint.com/automata_theory/regular_expressions.htm
- https://www.geeksforgeeks.org/python-regex/
- https://docs.streamlit.io/
- https://www.youtube.com/watch?v=ZZ4B0QUHuNc
- https://cs.utdallas.edu/people/faculty/
- https://www.regular-expressions.info/