**University of Arkansas – CSCE Department**

**Capstone II – Final Report – Spring 2020**

# PDF Extraction and Clean-Up

**Sarah Bondurant, Nathan Davis, Richard Mays, Keegan Riley, Hayden**

**Willeford**

## **Abstract**

Due to their consistency across software and devices, PDF files are one of the most used file formats in today’s academic world. PDF files are easily readable to humans, but machines can struggle to interpret the text. This project will use Natural Language Processing (NLP), computer vision, and human intervention to train a model that would convert PDFs to document files. The main goal is to restructure the text into a more readable and linear format to allow for further data processing.

## **1.0 Problem**

PDF files are one of the most used file formats in the modern world. Most online academic resources come in PDF form due to their consistency across different devices and software. They are easily readable by the user, and overall they look clean and professional. However, unlike their user-friendliness with humans, machines struggle to interpret them. This causes issues when machines try to read PDFs and creates inconsistencies in the machine's interpretation of them.

Parsing a PDF leads to unexpected results, ranging from mis-ordering text in columns to incorrect formatting. Correct parsing of columns would read down one column then down the next. However, with PDFs, sometimes text is read across the columns, garbling the data up. With math formulas in PDFs, equations that involve subscripts and superscripts might see them moved around to an incorrect position. Also, tables or other graphics can be placed in completely incorrect sections of text. All of these issues lead to incorrect documents in a corpus, causing untold complications for customers and the provider alike.

## **2.0 Objective**

The objective of this project is to research existing Natural Language Processing (NLP) and computer vision implementations and possibly new ideas to restructure a PDF’s text into a more easily readable structure for machines to interpret.

## **3.0 Background**

### **3.1 Key Concepts**

Two key technologies that are related to the problem and are essential to the development of the solution are NLP and computer vision. NLP, or Natural Language Processing, was developed to aid computers to understand the user's natural language. This being said, it is not an easy task to teach machines to be able to do this. NLP is a branch of AI that uses machine learning to take in input and learn how a language is used then being able to replicate it. Computer vision is a field that focuses on how computers are able to gain a high-level understanding from digital images or videos. Using both of these technologies, we could use machine learning to "teach" a computer to use its computer vision to run through documents and then use its NLP to then be able to extract and clean up the PDF.

### **3.2 Related Work**/Other Solutions

There are several public projects that work in PDF extraction using NLP and computer vision. These include XpdfReader, an open source PDF reader, Textricator, a tool that extracts text from documents, Apache PDFbox, an open source Java tool for PDF extraction, and PyPDF, a python based PDF library. Along with PARSR, these projects heavily explore text extraction from PDFs.

## **4.0** Parsr Results

### **4.1** What is Parsr?

Parsr is a public project available on GitHub. We were sent the information on Parsr from our industry contact. Parsr is only available to run in a Linux Ubuntu environment that has Docker installed as well. The program launches a GUI that can import and export PDF files to any file that could be requested. Now the GUI also has some added features that are called modules inside the code that sifts through the different files that are uploaded. These modules are different functionalities that can be used inside of Parsr. Those modules that are available range from a module that finds the whitespace between characters to one that finds the type of font being used. There are numerous different options.

List of requirements and/or Use cases:

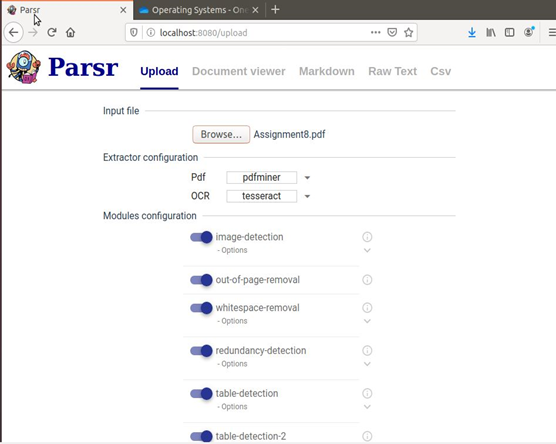
* Linux Ubuntu Environment
* Docker Engine
* Hard to access on a large team level but easy for an individual

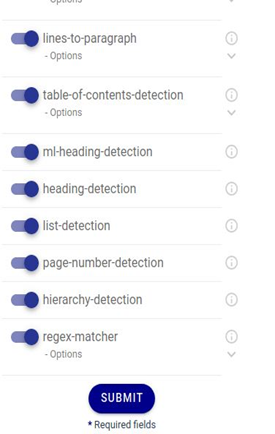
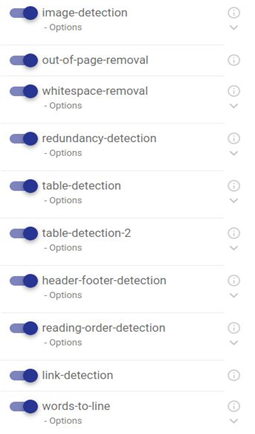
To Launch:

* Need to run the API for both the server and the GUI through terminal
* Then access to a browser

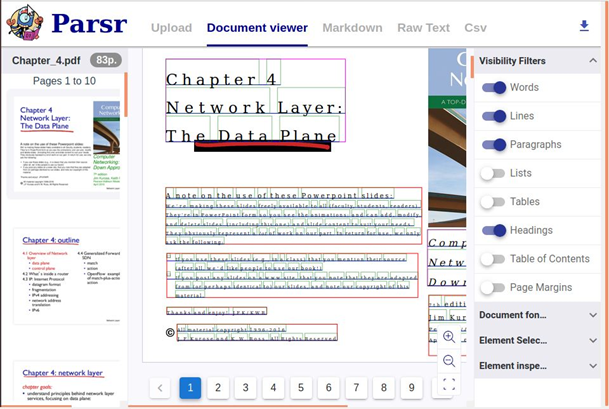
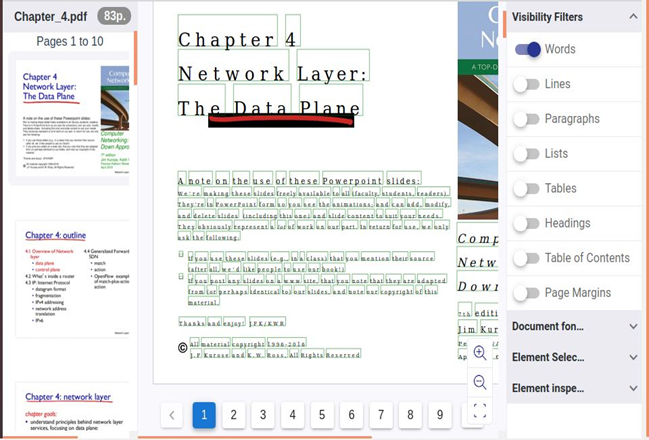
### **4.2** Interface

For this project and after looking into parsr and finding its limitations throughout the process, we realized that even though Parsr was the most robust, it was not perfect and at times hard to use.

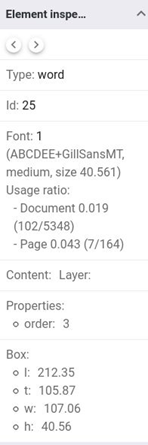
* Main Page -
  + In the main page, you are able to load the PDF file that you wish to extract and clean up as well as select your extractor and module configurations
* Module Information -
  + Modules are what make Parsr run. They are the code that is able to find the different parts of the PDF file and pull them apart to better sift through them. Some examples of the modules that are hereditary to Parsr are: image-detection, table-detection, and heading-detection.
  + Now the modules that are natural to Parsr are not the limitation of the application. You are able to add modules to a Parsr repository and then use them inside the GUI when launched.



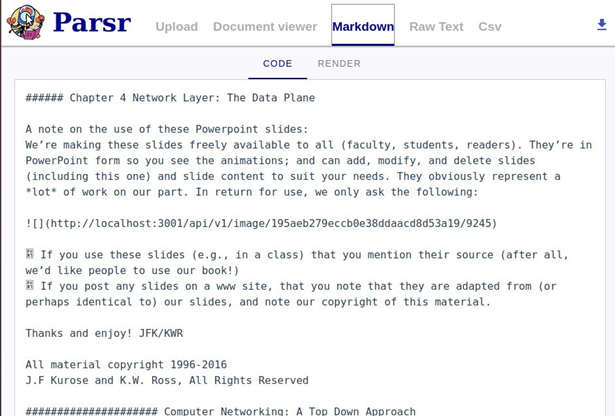
* Features -
  + Main Output
    - Inside the main output, you are able to see a panel that contains the PDF as if you had opened it inside a PDF reader, the main output that Parsr computes depending on the modules you choose, and the visibility filters that outline or highlight certain features of the text like words, lines, and paragraphs. In the second image, the green outline is for the words, the blue underline is the lines, the orange is the paragraphs, and the pink is the heading.



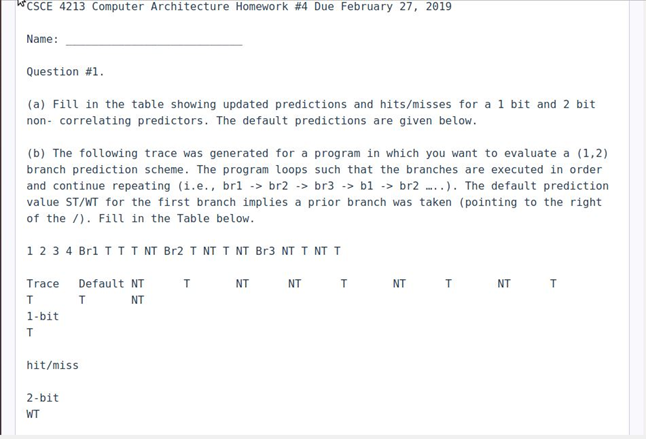
* + Element Inspection
    - You are able to select any of the objects or parts of the text and see the element information. This would include what type of object it is, the id, the font it is in, and multiple different properties.



* + Markdown Output
    - Parsr also outputs the file into a markdown version which would show the text in code form.



* + Raw Text
    - This is the last version of the output that Parsr outputs when you send in a file. This is the raw text of the file so Parsr is able to find the words in the PDF and output the text without any font or special adaptiations.



### **4.3** Pros / Cons

|  |  |
| --- | --- |
| **Con** | **Why Con?** |
| Required Linux Environment | Not a lot of computers run naturally on Linux |
| Required Docker Engine | Docker Engine would need to be installed |
|  |  |

### **4.4 Tasks**

1. Research NLP/Computer Vision
2. Integrate into Sorcero’s workflow
3. Researching promising PDF extraction implementations
4. Set-up & test Parsr
5. Familiarizing with Parsr codebase
6. Work on contribution to Parsr
7. Experiment with Python NLP
8. Continue experiments
9. Documentation write-up
10. Final Report & Presentation

### **4.5 Schedule**

|  |  |
| --- | --- |
| **Tasks** | **Dates** |
| 1. Research NLP/Computer Vision | 1/13 - 1/17 |
| 1. Integrate into Sorcero’s workflow | 1/21 |
| 1. Researching promising PDF extraction implementations | 1/20 - 2/1 |
| 1. Set-up & test Parsr | 2/3 - 2/15 |
| 1. Familiarizing with Parsr codebase | 2/17 - 2/29 |
| 1. Work on contribution to Parsr | 3/2 - 3/14 |
| 1. Experiment with Python NLP | 3/16 - 3/28 |
| 1. Continue experiments | 3/30 - 4/11 |
| 1. Documentation write-up | 4/13 - 4/18 |
| 1. Final Report & Presentation | 4/20 - 4/25 |

### **4.6 Deliverables**

* Web site - The website main page shows a project summary, task list/schedule, and links to other document deliverables. Individual team member pages are also found here.
* Final Report - The final report, this document, contains details about the project at length
* Final Presentation Video - This is the video presentation explaining our experiences and outcomes throughout the project
* Poster - The poster serves as a snippet to visually showcase the project as a whole in less detail than in the report/video

## **5.0 Key Personnel**

**Sarah Bondurant** – Bondurant is a senior Computer Science major in the Computer Science and Computer Engineering Department at the University of Arkansas. She has completed Programming Paradigms and Software Engineering. Over the past summer, she worked in web development. She will be responsible for UI design and the user guide.

**Nathan Davis** – Davis is a senior Computer Science major in the Computer Science and Computer Engineering Department at the University of Arkansas. He has completed Programming Paradigms and Software Engineering. He will be responsible for testing and documentation.

**Richard Mays** – Mays is a senior Computer Engineering major in the Computer Science and Computer Engineering Department at the University of Arkansas. He has completed Programming Paradigms, Software Engineering, and Computer Architecture. He will be responsible for creation of the initial prototype and the first prototype update.

**Keegan Riley** – Riley is a senior Computer Engineering major in the Computer Science and Computer Engineering Department at the University of Arkansas. He has completed Programming Paradigms and Software Engineering. He will be responsible for the second and third prototype updates.

**Hayden Willeford** – Willeford is a senior Computer Engineering major in the Computer Science and Computer Engineering Department at the University of Arkansas. He has completed Software Engineering. He will be responsible for generalizing the prototype for different fonts.

**Sorcero** – Sorcero is building an Enterprise NLP suite that supports the Life Sciences & Insurance industries in handling complex information, policies, rules, and regulations, automating the building of incredibly smart knowledge bases and workflows that rely on technical language.

## 6**.0 Reference**s

[1] <https://github.com/pdfpeople/Parsr>

[2] Wiriyathammabhum, Peratham & Summers Stay, Douglas & Fermüller, Cornelia & Aloimonos, Yiannis. (2016). Computer Vision and Natural Language Processing: Recent Approaches in Multimedia and Robotics. ACM Computing Surveys. 49. 1-44. 10.1145/3009906.

[3] <http://www.xpdfreader.com/index.html>

[4] <https://github.com/claird/PyPDF4>

[5] <https://github.com/measuresforjustice/textricator>

[6] <https://pdfbox.apache.org/>