# Automatic Document Formatting and Content Recognition for Academic Papers

Pranjul Mishra
Warsaw University of Technology
pranjul.mishra.stud@pw.edu.pl

Saurabh Singh
Warsaw University of Technology
saurabh.singh.stud@pw.edu.pl

Nazira Tukeyeva
Warsaw University of Technology
nazira.tukeyeva.stud@pw.edu.pl

**Supervisor: Anna Wróblewska**
Warsaw University of Technology
anna.wroblewska1@pw.edu.pl

## Abstract

The exponential growth of academic literature has increased the demand for automated tools to manage and organize large volumes of unstructured data. The "Automatic Document Formatting and Content Recognition for Academic Papers" project proposes a system that automates the recognition and extraction of key components from research documents. By leveraging advanced NLP and information extraction techniques, this system identifies essential elements such as titles, authors, and structured content. This project aims to improve document management workflows, making literature analysis more efficient for researchers.

## 1 Introduction

The rapid growth of scientific literature has created an overwhelming amount of unstructured data, requiring researchers to spend considerable time manually extracting essential elements like titles, authors, and main sections from academic documents. This project addresses this challenge by developing a system to automatically recognize and process the format of academic documents, specifically research papers and articles, to extract key textual and non-textual components.

The system will analyze documents in PDF and DOCX formats, extracting sections such as title, author(s), abstract, introduction, and conclusion, as well as embedded tables and images. By automating this process, the system aims to reduce the manual workload for researchers and enhance the efficiency of literature review and content management tasks.

### 1.1 Research Questions

This project focuses on addressing the following key research questions:

1. How can we design a method to automatically detect and extract essential components from academic documents (e.g., title, author(s), abstract, and section content)?
2. Which NLP techniques are most effective for segmenting document content by headings and subheadings?
3. How can non-textual elements, such as tables and images, be accurately identified and extracted from academic documents, given the complexity of formats like PDFs?

### 1.2 Project Goal

The primary goal is to create an automated system that processes academic documents and extracts:

- **Titles** and **author(s)** with affiliations,
- **Section content** organized by headings (e.g., abstract, introduction, results),
- **Non-textual elements** such as tables and images.

## 2 Concept and Work Plan

The project is structured into three main phases across a 10-week period:

### 2.1 Work Plan

The project will proceed in three structured phases:

**Phase 1: Project Proposal (Weeks 1-2)** focuses on finalizing the project proposal, conducting a literature review, and setting clear objectives and milestones.

**Phase 2: Proof of Concept (Weeks 3-7)** involves developing a minimal viable product (MVP) that identifies document structures and extracts essential components, like titles and authors, from diverse documents.

**Phase 3: Final Project (Weeks 8-10)** includes refining the MVP for accuracy and robustness, completing the user interface, and finalizing the project report and presentation.

### 2.2 Risk Analysis

Key challenges include: **Inconsistent Document Structures**, which require flexible NLP models to handle varied formats, and **Non-Textual Element Extraction**, where the complex layouts of PDFs can make extracting tables and images difficult. Using advanced NLP and computer vision techniques aims to mitigate these risks.

## 3 Open Dataset Review

To evaluate our model and benchmark its performance, we will utilize several open datasets that are widely used in the field of document processing and content extraction. Each dataset provides distinct types of structured content, allowing us to rigorously test our system's ability to handle various document layouts and extraction tasks.

- **PubMed Central Open Access Subset (PMC-OAS)** [1]: The PMC-OAS is a comprehensive dataset containing millions of open-access biomedical and life sciences research articles. It offers both PDF and XML formats, with the XML format containing structured metadata and clearly labeled sections, such as titles, authors, abstracts, and main body text. This dataset is particularly valuable for testing the accuracy of title, author, and section extraction due to its well-annotated and structured format. Since the biomedical field involves complex documents with structured figures and tables, PMC-OAS will enable us to evaluate our system's capabilities in processing and extracting such non-textual elements effectively.

- **arXiv Dataset** [2]: The arXiv dataset is a large repository of open-access scientific papers covering a broad range of fields, including computer science, physics, and mathematics. Available in multiple formats, including PDFs, arXiv papers often exhibit diverse structural characteristics, with varying layout styles, section headings, and citation formats. This diversity makes arXiv an ideal dataset for testing the model's adaptability to different document structures and for training the system to handle various formats encountered in academic literature. Additionally, arXiv's extensive coverage across fields will allow us to assess how well the system generalizes across domains, as well as how effectively it extracts complex mathematical and technical content often present in these papers.

- **ICDAR Competition Datasets** [3]: The ICDAR (International Conference on Document Analysis and Recognition) competition datasets are specifically curated for evaluating document layout analysis and content extraction models. These datasets provide a range of document types, including research papers, technical reports, and scanned documents, along with ground truth labels for structured metadata, sections, and layout information. ICDAR datasets serve as a benchmark for document structure recognition and segmentation, as they contain both printed and scanned documents, challenging our model to accurately extract information despite noise and format inconsistencies. Evaluating our system on ICDAR datasets will help validate its robustness and accuracy in processing varied document layouts, especially in noisy or less-than-ideal document conditions.

- **GROBID (GeneRation Of BIbliographic Data)** [4]: GROBID is an open-source tool and dataset focused on extracting structured bibliographic metadata from scientific documents. Widely used in NLP and information extraction benchmarking, GROBID provides labeled data for extracting titles, authors, affiliations, publication dates, and references. This dataset is particularly valuable for testing and fine-tuning the metadata extraction component of our system, as GROBID's data covers various types of documents with different citation and metadata formats. Testing against GROBID's structured data will allow us to measure our model's effectiveness in handling bibliographic content, a crucial feature for applications in academic search engines and citation management systems.

By leveraging these datasets, our system will undergo rigorous testing across various document structures, formats, and content types. This diversity will provide comprehensive insights into the model's strengths and limitations, informing iterative improvements for handling both well-structured documents and those with variable or complex layouts.

## 4 Methodology

Our approach integrates Natural Language Processing (NLP), machine learning, and image processing techniques to develop a robust, automated system for document formatting and content recognition. This methodology section outlines the key steps, tools, and evaluation metrics that will guide the development and validation of the system.

### 4.1 Data Collection and Preprocessing

To train and evaluate the system, we will collect academic documents from open-access repositories such as arXiv, PubMed, and other public sources. These repositories provide a diverse array of documents with varied formats, including PDFs and XML. Preprocessing these documents involves several key steps:

- **Text Extraction**: We will use PDFMiner for extracting raw text from PDF files, which serves as the foundation for further analysis.
- **Tokenization and Segmentation**: The extracted text will be tokenized and segmented to facilitate content recognition. Tokenization breaks down the text into individual words or phrases, while segmentation helps divide the document into distinct sections based on headings and subheadings.
- **Noise Removal**: To enhance data quality, we will remove unnecessary elements, such as page numbers, footnotes, and formatting artifacts, that do not contribute to the document's informational structure.

## 4.2 Document Segmentation and Content Extraction

Accurate segmentation and content extraction are essential to recognize document structure and identify specific components. To achieve this, we will employ advanced NLP models and Named Entity Recognition (NER) techniques:

- **NLP Models**: Transformer-based models like BERT [6] will be employed to understand and segment the document based on contextual relationships within the text. BERT's pre-trained language representations provide a strong basis for recognizing key sections such as the title, authors, and abstract.
- **Named Entity Recognition (NER)**: NER will be used to identify entities like author names, affiliations, and dates, which are critical components of academic documents. By training the model to recognize these specific entities, we enhance its ability to accurately capture and categorize information.
- **Regular Expressions**: To support the NLP models, we will use regular expressions for identifying commonly formatted sections, such as references or bibliography, ensuring consistency across documents with standardized headings.

## 4.3 Non-Textual Element Extraction

Extracting non-textual elements such as tables and images is a critical component of this project, as these elements carry valuable information in academic documents. Our approach to non-textual element extraction combines Optical Character Recognition (OCR) and layout analysis:

- **OCR for Embedded Text**: We will utilize Tesseract, an OCR tool, to extract text from images, which is essential for identifying text within figures or diagrams.
- **Table Extraction Tools**: Tools like Camelot [8] and Tabula [9] will be used to detect and extract tabular data from PDF files. These tools are well-suited for handling structured data and can accurately retrieve content from tables with standard grid layouts.
- **Layout Processing with OpenCV**: To recognize and isolate images and figures, OpenCV will be employed to analyze layout structure, detect visual boundaries, and separate non-textual elements from the main body text. This will facilitate accurate extraction and ensure the preservation of document layout.

## 4.4 Evaluation and Benchmarking

The performance of our system will be rigorously evaluated using a set of standardized metrics to ensure accurate and reliable results. These metrics will provide quantitative insights into the system's effectiveness in content extraction and layout recognition:

- **Precision, Recall, and F1-score**: These metrics will be calculated for key components such as title, author, and section extraction, providing a comprehensive view of the model's accuracy.
- **Intersection over Union (IoU)**: IoU will be used to evaluate the accuracy of non-textual element extraction, particularly for tables and images, by measuring the overlap between detected and ground truth elements.
- **End-to-End Processing Time**: The system's efficiency will be assessed by measuring the total processing time required for document parsing, which is essential for scalability and practical usability.

Human evaluators will further assess the quality of extracted data to provide qualitative feedback on the system's real-world application.

## 4.5 Plans for Comparison with State-of-the-Art Solutions

To ensure our system meets or exceeds current standards, we will benchmark it against state-of-the-art solutions in document analysis and content extraction, focusing on methods like BERT, LayoutLM, and GROBID:

- **BERT (Bidirectional Encoder Representations from Transformers)**: Known for robust text segmentation, BERT will serve as a strong baseline, particularly for text-based content extraction tasks [6].
- **LayoutLM**: LayoutLM integrates both textual and spatial information, making it highly effective for understanding complex document layouts like

PDFs [7]. This comparison will highlight the effectiveness of layout-aware models relative to our approach.

- **GROBID (GeneRation Of BIbliographic Data)**: GROBID focuses on bibliographic data extraction, making it a valuable benchmark for evaluating metadata extraction, such as titles, authors, and affiliations [5].

We will compare our model against these methods using the same datasets and evaluation metrics to establish a fair and rigorous benchmark. This comparative analysis will help identify areas where our model excels or requires improvement, ensuring that it is competitive with the latest advancements in the field of document processing.

# 5 Literature Review

This literature review examines recent advancements in NLP and document processing, focusing on document structure recognition, content segmentation, non-textual element extraction, and the challenges associated with processing diverse document layouts.

## 5.1 Document Structure Recognition

Accurately recognizing document structures is essential for extracting meaningful information from academic papers, which typically follow standardized formats. Early systems, such as CERMINE [5], pioneered automated metadata extraction and document segmentation by combining rule-based methods and machine learning. While effective for well-structured documents, these methods often struggled with inconsistent formats, limiting their generalizability.

With the advent of transformer models, document structure recognition has significantly advanced. Models like BERT (Bidirectional Encoder Representations from Transformers) [6] have demonstrated a remarkable ability to capture contextual relationships within documents, making them highly effective for text segmentation tasks. Building upon BERT, LayoutLM [7] integrates both textual and spatial information, enabling it to recognize complex document layouts, such as those found in PDFs, by leveraging positional embeddings to understand content location. These transformer-based models outperform traditional methods by accurately identifying document structures across varied layouts, thus providing a more robust solution for academic document processing.

## 5.2 Content Segmentation Techniques

Content segmentation plays a critical role in dividing a document into its constituent sections (e.g., abstract, introduction, conclusion) and is foundational for structured information extraction. Traditional methods for segmentation relied heavily on predefined templates or rule-based approaches [14]. Although useful for highly standardized documents, these methods lack the flexibility needed to adapt to varying formats and heading structures across different types of documents.

Recent advancements in neural networks have introduced more flexible and adaptable segmentation techniques. Neural sequence labeling models, such as the one proposed by [10], apply recurrent neural networks (RNNs) to segment text by learning contextual patterns within document content. Transformer models, which can capture long-range dependencies and hierarchical relationships within text, have further improved segmentation robustness. By leveraging attention mechanisms, transformers can accurately segment documents even when section headings are ambiguous or formatted inconsistently. This adaptability makes them well-suited for academic documents, which often exhibit variability in structure and layout.

## 5.3 Non-Textual Element Extraction

The extraction of non-textual elements, such as tables, figures, and images, presents unique challenges due to the varied ways these elements are embedded within documents. Non-textual elements are especially prevalent in scientific literature, where tables and figures convey critical information. DeepDeSRT [11] introduced a deep learning approach specifically for table detection and structure recognition in document images, marking a significant advancement over traditional image processing techniques. By utilizing convolutional neural networks (CNNs), DeepDeSRT can detect tables based on their visual characteristics, making it effective for table-heavy academic documents.

Additionally, tools like Tabula [9] and Camelot [8] have been developed for extracting tabular data from PDFs, facilitating the retrieval of structured information. These tools are valuable for simple table layouts, but complex tables with merged cells, spanning rows or columns, still pose challenges and may require further refinement. Combining these tools with layout analysis frameworks, such as OpenCV, enhances the ability to detect and extract non-textual elements, ensuring that tables, images, and figures are accurately preserved in the extracted data.

## 5.4 Challenges in Document Analysis

The diversity in document layouts remains a significant hurdle in developing a generalized system for document analysis. Academic documents vary widely in terms of formatting styles, heading structures, and the placement of non-textual elements, all of which can impact the performance of document processing systems. Antonacopoulos et al. [12] emphasize the importance of realistic datasets for evaluating document layout analysis, noting that many existing models struggle with real-

world document variability.

Another challenge is the limited availability of large, annotated datasets for training document analysis models. The ICDAR competition dataset [13] highlights this limitation, as it is one of the few resources providing labeled data specifically for table detection and recognition tasks. Without sufficient annotated datasets, models may lack the robustness needed to generalize across different document types, leading to inconsistent results when applied to unseen document formats.

### 5.5 Our Contribution

Our project aims to address these challenges by developing a system that integrates transformer models with advanced processing tools tailored for academic document analysis. By leveraging transformer-based models like BERT and LayoutLM for document segmentation and NER, our approach aims to achieve accurate recognition of both textual and spatial information. Additionally, our use of specialized tools, such as Camelot and Tabula for table extraction and OpenCV for image processing, will facilitate comprehensive extraction of non-textual elements, ensuring that key information in tables, images, and figures is preserved. Through rigorous evaluation and benchmarking, our system aspires to offer a robust, adaptable solution for automated academic document processing.

## 6 Proof Of Concept and Preliminary Report

The rapid growth of scientific literature has created a pressing need for automated tools capable of processing and analyzing academic documents. This project focuses on developing a system for recognizing and extracting structured content from academic papers, particularly research articles. The goal is to enhance document processing workflows by automating tasks such as title and metadata extraction, section segmentation, and hierarchical structuring.

The system processes documents in PDF format (with DOCX support planned for future phases) and extracts structured data, including the title, authors, metadata, main sections, and non-textual elements such as tables and images. This report outlines the progress achieved so far and presents the proof of concept (POC) for the modules developed.

### 6.1 Objectives

The primary objectives of the project are as follows:

1. Automatically detect the format of the document (PDF/DOCX).
2. Extract raw text and layout metadata from the document.
3. Preprocess the text by cleaning, tokenizing, and segmenting it for structured analysis.
4. Recognize and hierarchically segment the document's structure into sections like title, abstract, authors, and main sections.
5. Enable Named Entity Recognition (NER) to extract metadata such as authors and affiliations.
6. Lay the groundwork for future modules, including the extraction of tables and images.

### 6.2 Architecture Design

The architecture of the proposed system follows a modular design, ensuring scalability and ease of integration. Each module is developed independently, allowing for iterative testing and refinement. The architecture diagram is shown in Figure 4.

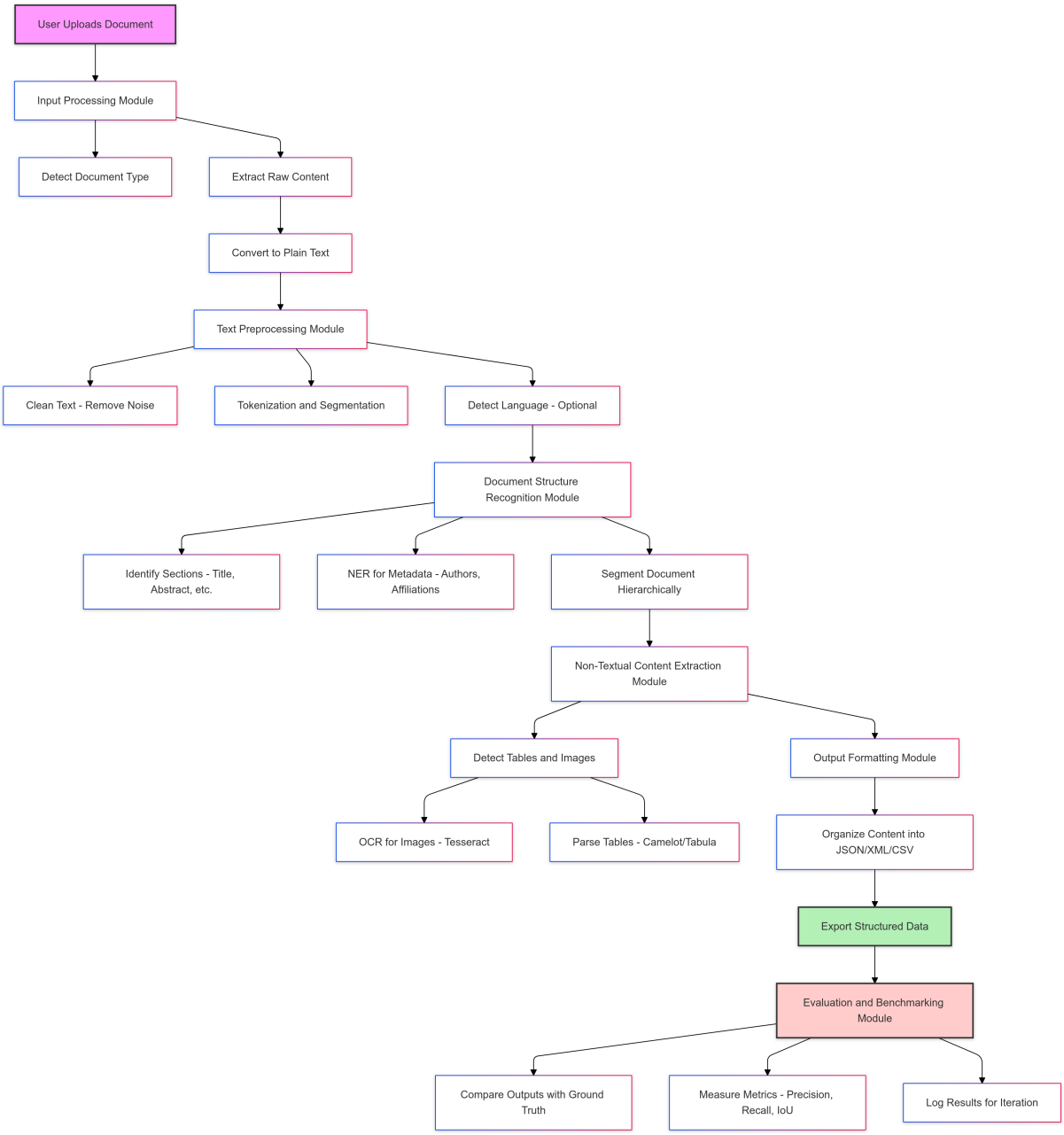


Figure 1: System Architecture

## 6.3 Progress Achieved

### 6.3.1 Overview of the Modular Architecture Design:

We were supposed to make significant changes in our approach inorder to improvise our performance efficiency and dependencies from different models and try a raw approach which could be further optimized to perform very fastly. This program is an all-in-one pipeline for extracting textual and visual elements from a PDF and organizing them into a structured JSON format . Specifically, it:

1. Renders each PDF page to an image using pypdfium2.
2. Applies OCR (via Tesseract) to extract text from each rendered page-image.
3. Identifies a line (heuristically deemed "most bold") on the first page to use as the title.
4. Captures the remaining text lines into a simple content list.
5. Extracts tables using Camelot (both in lattice and stream modes).
6. Extracts embedded images using PyMuPDF (fitz) and stores them in an output folder.
7. Saves everything (text, tables, images) as a JSON object, for further processing or consumption.

Let us look at it in more detail what functions are doing at each step in order to obtain a better understanding.

### 6.3.2 Converting PDF to Images (Input Processing Module)

1. **Function:** `convert_pdf_to_images (pdf_path, dpi=300)`
2. **Library:** `pypdfium2`
3. **Mechanism:**
   - Opens the PDF with `pdfium.PdfDocument(pdf_path)`.
   - Iterates over every page.
   - Renders each page into a PIL image at 300 DPI
     (higher DPI yields better OCR results).
   - Each rendered image is converted to an in-memory
     JPEG (bytes) and appended to a list as {`page_index: image_bytes`}.

Because Tesseract performs best on clear, high-resolution images, we set `dpi=300` to optimize image clarity for OCR.

### 6.3.3 Running OCR on Each Page (Text Preprocessing Module)

1. **Function:** `extract_text_from_pdf_with _tesseract(list_of_image_dicts)`
2. **Library:** `pytesseract`
3. **Mechanism:**
   - Receives a list of dictionaries, each containing
     {`page_index: image_bytes`}.
   - Sorts the list by `page_index` to maintain consistent page order.
   - For each page-image dictionary:
     - Converts bytes back into a PIL image.
     - Calls `pytesseract.image_to _string()` to get a raw text string.
   - Returns a list of page-level texts (one string per page).

### 6.3.4 Identifying a Title and Structuring Text (Document Structure Recognition Module)

1. **Function:** `structure_text_into_json (page_texts)`
2. **Title Detection:** The code calls `find_first_bold_line(...)` on the first page's lines.
   - It checks each line's length and uppercase ratio.
   - The first line that meets a threshold is labeled as the main title.
   - If no lines qualify, it defaults to `"Unknown Title"`.
3. **Paragraph Collection:** Concatenates all page lines (except the recognized title line) into a content list of paragraphs.

### 6.3.5 Table Extraction (Document Structure Recognition Module: Non-textual Elements)

1. **Function:**`extract_tables(pdf_path, output_dir="tables_output")`
2. **Library:**`Camelot`
3. **Mechanism:**
   - Attempts `lattice` mode: Suited for PDFs with lines forming clear table structures.
   - Attempts `stream` mode: More tolerant of text-based tables without explicit bounding lines.
   - Each found table is saved as a CSV file in `tables_output/`.
   - Returns a list of references (page number, CSV path, flavor).

#### 6.3.6 Embedded Image Extraction (Document Structure Recognition Module: Non-textual Elements)

1. **Function:** `extract_images(pdf_path, output_dir="images_output")`
2. **Library:** `PyMuPDF (fitz)`
3. **Mechanism:**
   - Opens the PDF with `fitz.open(...)`.
   - For each page, calls `page.get_images(full=True)` to list all embedded images.
   - Each extracted image is written to disk (e.g., `page1_img1.png`) in `images_output/`.
   - A metadata list is returned, referencing each saved file and which page it came from.

#### 6.3.7 Output JSON Assembly (Exporting Structured Document)

1. **Function:** `parse_pdf_with_pypdfium_tesseract_tables_images(pdf_path, output_json="final_output.json")`
2. Converts PDF to images.
3. Performs OCR on those images to extract text.
4. Identifies a title and merges the rest of the text into a content array.
5. Extracts tables and images.
6. Builds a final JSON dictionary:

```
{
  "title": "...",
  "pages_ocr": [
  "Paragraph1...",
  "Paragraph2...", ...],
  "tables": [
    {"page": X,
    "csv_path": "...",
    "flavor": "..."},
    ...
  ],
  "images": [
    {"page": X,
    "file": "..."},
    ...
  ]
}
```

7. Writes the resulting JSON to `final_output.json`.

# 7 Results

## 7.1 Text Extraction

We successfully extracted text from the provided PDF file in a structured format, enabling a clear representation of the document's content. This structured output can be effectively utilized to understand the data and even serve as a foundation for training a Large Language Model (LLM). However, the dataset presented a significant challenge due to inconsistencies in its formatting across different files.To address this variability, we experimented with multiple data formats to accommodate the diverse structures encountered. Despite our efforts, establishing a universal hierarchical structure proved challenging because of the dataset's lack of uniformity. This inconsistency prevented us from defining a single, definitive structure that could seamlessly represent the hierarchy across all documents.

Nonetheless, the methodologies applied provide a robust foundation for understanding the underlying structure of the data, and further refinements could lead to improved adaptability to varying document formats.

## 7.2 Image Extraction

We sucessfully extracted the images from the Academic papers with high quality in a folder named as "images output" in the same directory and in ready to use format which included the logo image, graphs in the academic paper etc. A few example images from an academic paper are as follows:

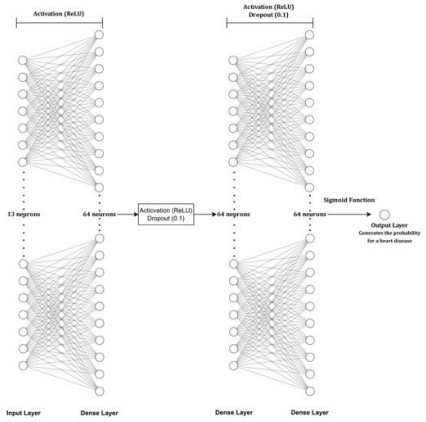


Figure 2: Example Extracted Image 1 from dataset

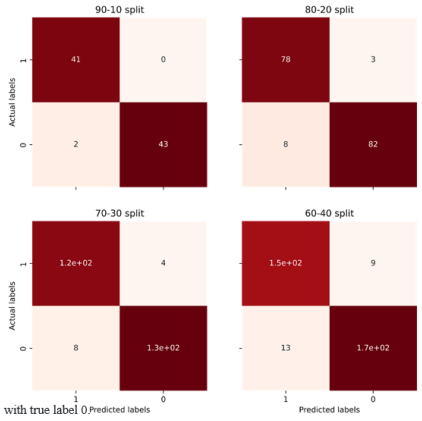


Figure 3: Example Extracted Image 2 from dataset

Images obtained were 100% accurate we didn't missed any single image from any academic paper so far, however if the format is very much unstructured, this could cause a certain scenarious where one or two images can be missed not more than that.

## 7.3 Table Extraction

We successfully extracted tables from the documents and saved them in the "Table Output" folder within the same directory. However, this process revealed certain limitations. In some instances, particularly with medical papers, references and image labels were mistakenly identified as tables . This misclassification arises from the algorithm's interpretation of certain document formatting, which can lead to inaccuracies.

The inconsistency in document formatting further exacerbates this issue, making it difficult to consistently differentiate between actual tables and other structured elements, such as reference lists. These limitations have been detailed in the "Challenges" section of this report, where we have outlined specific cases and scenarios in which the algorithm may fail to accurately extract tables.This acknowledgment is intended to provide clarity on the constraints of the current methodology and to inform future enhancements.

tables_output > table_page9_stream_14.csv > data

```
0,1,2,3,4,5
,,,to employ the Bootstrap Aggregation with Random Forest.,,
"F
ig. 17: ROC curve of deep neural network model history",,
,technique which is illustrated in Table 6.,,
"T
able 5: Performance of all the models",,,,,
Model,Split,Accuracy,Cohen Kappa,R2 Score,AUC
Logistic Regression,90-10,0.95348,0.90677,0.81355,0.9533
,80-20,0.94736,0.89450,0.78888,0.9475
,70-30,0.95719,0.91434,0.82835,0.9580
,60-40,0.94152,0.88255,0.76492,0.9416
Support Vector Machine (SVC),90-10,0.91860,0.83667,0.67371,0.9178
,80-20,0.92982,0.85908,0.71851,0.9290
,70-30,0.89105,0.78067,0.56308,0.8887
,60-40,0.88888,0.77519,0.55335,0.8850
MLP with PCA,90-10,0.91860,0.83703,0.67371,0.9189
```

Figure 4: Example Extracted Table from dataset

This is an example of how tables are extracted from the pdf files and stored seperately in CSV format.

## 7.4 Comparision With Existing State of the Art Tools

Our Method is undoubtedly fast and optimized but it lacks accuracy in some cases and compared to other state of the Art Models we could perform on a decent level only.

- Compared with GPT4 Latest model our model was accurate in terms of image extraction, however in table and text extraction, our approach was decent in terms of structured output.
- Compared with GROBID as sophisticated tool, We observed that image and text extractions were significantly perfect to be compared, however tables and structured output of the text was lacking.
- In comparison with GPT3-Turbo model we did perform very well, except for structuring, since the model is heavily trained however in table and image scenario our approach was better.

# 8 Challenges

### 8.0.1 Limited Multi-Column Accuracy

- Tesseract often merges text across columns. Even though a higher DPI improves OCR results, truly multi-column PDFs may produce interleaved lines in the OCR output.
- **Example:** A 2-column research paper might have lines from the right column appear in the middle of the left column's text.
- This implementation does not include advanced column separation, which limits its accuracy for multi-column layouts.

### 8.0.2 "Bold Line" Heuristic

- The code uses uppercase ratio and line length to identify the "bold" line as the title. If the actual bold title is not uppercase or is relatively short, it may not be detected accurately.
- **Example:** A title with many lowercased words or a short, stylized title might fail to meet the "uppercase ratio" threshold and thus be missed or partially detected.

### 8.0.3 Non-Textual or Complex Tables

- Camelot is designed for text-based tables. If the table is a scanned image or has a highly complex structure, the extraction process may fail.
- While the code logs a warning in such cases, it does not implement OCR-based table recognition as an alternative.

- **Example:** A table embedded as a graphic or containing complicated merged cells may not produce any CSV output.

#### 8.0.4 LLM Integration

As mentioned in presentation we tried experimenting with "T5-small" and "Bert-based -Uncased" models from Hugging face which are significantly trained on some rule based approach to handle extraction, they did not performed well in our case, instead our approach was much faster and optimized. We are somewhat biased to conclude in this case as this could mostly due to system's hardware limitation but efficiency of the program was our priority.

# 9 Future Work

1. **Output Formatting**: The extracted content could be organized into structured formats such as JSON, XML, or CSV. These formats can then be converted into sets of triples using predefined rules, facilitating the creation of a knowledge graph.

2. **Advancements in LLMs**: Integration with more advanced large language models (LLMs) could improve the process. To be precise, this would reduce potential nuances in the extraction process, while enhancing the hierarchical organization of data when dealing with multiple PDF files simultaneously.

3. **Enhanced Capabilities**: A more efficient approach could be developed to handle complex formulas, improving the accuracy and scalability of the system.

#### 9.0.1 Advanced Layout and Column Detection

- Incorporating a step that splits each rendered page-image into left/right halves for Tesseract (if multi-column layout is confirmed).

- Integrating a layout analysis approach that identifies bounding boxes for each column to improve text segmentation was quite sophisticated.

#### 9.0.2 Better Metadata and Title Extraction

- Current implementation relies on a simple uppercase ratio heuristic.

  - Utilizing PDF text bounding boxes or font size variations.
  - Leveraging specialized metadata extraction tools, such as GROBID, to reliably detect article titles, authors, and affiliations.

#### 9.0.3 OCR-Based Table Extraction and Intelligent Merging

For tables that fail Camelot due to scanning or complex merges, an OCR-based approach or advanced "table recognition" solution could be considered. Integrating an additional pass for scanning potential table regions could significantly improve table results could be done.

## 9.1 Reviews

This chapter addresses the feedback provided by reviewers. Each point is listed along with our response to ensure clarity in addressing the suggestions for our project.

**1. Benchmarking Performance**

**Feedback:** Include a robust testing framework for real-world scenarios, such as scanned documents or those with low-quality images.

**Answer:** The performance is being validated using ICDAR and GROBID datasets, which include noisy and scanned documents. Additionally, integrating OCR tools like Tesseract enhances the processing of low-quality images.

**2. Non-Textual Element Extraction**

**Feedback:** While the system uses tools like Camelot and Tabula for table extraction, it might struggle with complex layouts (e.g., merged cells, multi-column tables).

**Answer:** As said above, as for the table extraction we rely on Camelot and Tabula, which are effective for standard layouts. However, we recognize their limitations with merged cells and irregular structures. Advanced tools like DeepDeSRT and layout analysis via OpenCV are explored to address these cases.

**3. Data Exploration and Description**

**Feedback:** Provide a detailed exploration of the dataset, including examples and demonstrations of the system's performance.

**Answer:** Section 4.5 has been dedicated to showcast utilized datasets, including PubMed, arXiv, ICDAR and GROBID datasets. Performance results with visual demonstrations can also be visible in "Results" section.

**4. Support for Other Document Types**

**Feedback:** Expand the system to handle additional formats like LaTeX, HTML, and EPUB.

**Answer:** Within the framework of NLP course, two main formats such as PDF and DOCX were chosen. Broadening usability across other academic formats is the planned study of future improvements.

**5. Improved Metadata Extraction**

**Feedback:** Enhance metadata extraction to include funding sources, keywords, and publication dates.

**Answer:** Named Entity Recognition (NER) pipeline is used to identify entities like author names, affiliations, and dates. Besides, fine-tuning SpaCy-based NER models enable these enhancements.

**6. Literature Review**

**Feedback:** Include recent advancements in transformer models, and discuss challenges of document analysis.

**Answer:** The literature review is updated to include advancements in OCR tools like Tesseract. Additionally, there is a review on recent publications regarding diversity in document layouts (formatting styles, heading structures), discussed in section 5.4.

### 9.2 Work Distribution

| Task | Assigned TM(s) |
|---|---|
| Project Proposal Report | All |
| PoC + Presentation 1 | All |
| Data Collection + Preprocessing | Pranjul |
| Document Segmentation | Nazira |
| Content Extraction | Saurabh, Pranjul |
| Non-Textual Element Extraction | All |
| Evaluation | Nazira, Saurabh |
| Benchmarking | Pranjul |
| Literature Review | All |
| Final Deliverables | All |

Table 1: Work Distribution

## 10 Conclusion

The progress so far demonstrates the feasibility of automating document processing for academic research papers. The POC successfully integrated modules for input processing, text preprocessing, and document structure recognition, forming a strong foundation for further development. Future work will focus on non-textual content extraction and robust evaluation. This project addresses the growing need for automated tools capable of handling the complex formatting and structure of academic documents, specifically those in PDF format. By developing a system that integrates advanced NLP models, machine learning, and image processing techniques, we aim to achieve accurate extraction of both textual and non-textual elements from research papers and articles. This system will facilitate the automatic recognition and organization of key document components, including titles, authors, sections, tables, and images, making it a comprehensive solution for academic document processing.

Leveraging a range of open datasets, such as PubMed Central, arXiv, and the ICDAR competition datasets, we will rigorously test and benchmark our system's performance. These datasets provide a diversity of document structures and content types, allowing us to validate the model's adaptability and robustness across various academic disciplines. The benchmarking process will include comparisons with state-of-the-art solutions like BERT, LayoutLM, and GROBID, ensuring our system meets or exceeds current standards in document analysis.

This project's outcomes have the potential to significantly streamline literature review and content extraction workflows in academic and research settings, reducing the manual effort required to process large volumes of scientific documents. Furthermore, by setting a solid foundation with structured methods for both text and non-text element extraction, this project paves the way for future advancements in automated academic document processing. Future work may build on this system's modular design, allowing for the integration of additional document types, such as technical reports or dissertations, and further enhancing the system's capabilities with emerging NLP and document processing technologies.

## References


[1] National Center for Biotechnology Information. "PubMed Central Open Access Subset." Available: `https://www.ncbi.nlm.nih.gov/pmc/tools/openftlist/`

[2] Cornell University. "arXiv Dataset." Available: `https://www.kaggle.com/Cornell-University/arxiv`

[3] ICDAR. "ICDAR Competition Datasets." Available: `https://tc11.cvc.uab.es/datasets/ICDAR2019cTDaR`

[4] "GROBID: GeneRation Of BIbliographic Data." Available: `https://github.com/kermitt2/grobid`

[5] Tkaczyk, Dominika, et al. "CERMINE: automatic extraction of structured metadata from scientific literature." *International Journal on Document Analysis and Recognition (IJDAR)* 18.4 (2015): 317-335.

[6] Devlin, Jacob, et al. "BERT: Pre-training of deep bidirectional transformers for language understanding." *arXiv preprint arXiv:1810.04805* (2019).

[7] Xu, Yang, et al. "LayoutLM: Pre-training of text and layout for document image understanding." *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2020.

[8] "Camelot: PDF Table Extraction for Humans." (2019). Available: `https://camelot-py.readthedocs.io/`

[9] "Tabula: Extract Tables from PDFs." (2020). Available: `https://tabula.technology/`

[10] Yang, Zhilin, et al. "Neural machine translation with recurrent attention modeling." *arXiv preprint arXiv:1703.04675* (2017).

[11] Schreiber, Sebastian, et al. "Deepdesrt: Deep learning for detection and structure recognition of tables in document images." *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*. Vol. 1. IEEE, 2017.

[12] Antonacopoulos, Apostolos, et al. "A realistic dataset for performance evaluation of document layout analysis." *2015 13th International Conference on Document Analysis and Recognition (ICDAR)*. IEEE, 2015.

[13] Gao, Liangcai, et al. "ICDAR 2019 competition on table detection and recognition (cTDaR)." *2019 International Conference on Document Analysis and Recognition (ICDAR)*. IEEE, 2019.

[14] Liu, Yang, et al. "Text segmentation by combining generative and discriminative methods." *Proceedings of the national conference on artificial intelligence*. Vol. 22. No. 2. 2007.

[15] Adhikari, Ashutosh, et al. "DocBERT: BERT for Document Classification." *arXiv preprint arXiv:1904.08398* (2019).

[16] Lafferty, John, Andrew McCallum, and Fernando Pereira. "Conditional random fields: Probabilistic models for segmenting and labeling sequence data." *Proceedings of the Eighteenth International Conference on Machine Learning*. 2001.