Incorporating Patient Preference Studies Into Clinical Research and Decision Models

Manuel Limideiro
Politecnico di Torino
Torino, Italia
s303529@studenti.polito.it

Shaoyong Guo
Politecnico di Torino
Torino, Italia
s296966@studenti.polito.it

Fabio Rizzi
Politecnico di Torino
Torino, Italia
s308770@studenti.polito.it

ABSTRACT

Extracting data from scientific publications, such as Patient Preference Studies (PPS), often involves a tedious manual process due to the unstructured nature of PDF documents. This paper presents an innovative automated system designed to streamline this process. Our solution integrates three core models: PDFigCapX for detecting images, ChartOCR for extracting data from charts, and Table Transformer for handling tabular data. This automation significantly reduces the time required for data extraction from PPS papers, empowering researchers to focus on higher priority work, to experiment with larger datasets and conduct advanced analytics. We discuss performance metrics, limitations, and propose directions for future development. Our GitHub repo can be found at the following link: https://github.com/Guo-SY/Incorporating-Patient-Preference-Studies-Into-Clinical-Research-and-Decision-Models

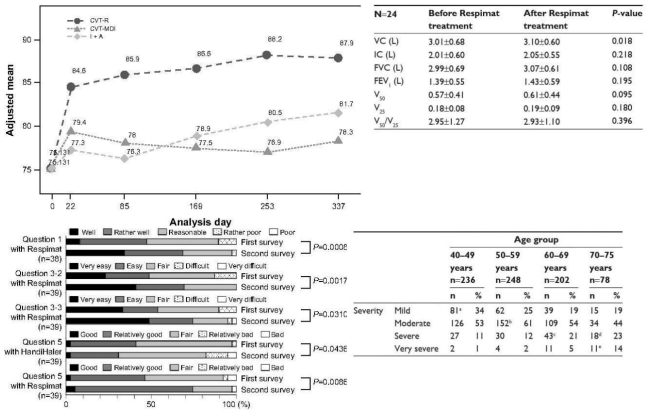

Figure 1: Examples of chart and table images contained in PPS papers.

1 INTRODUCTION

Patient preference studies are a type of medical research that aims to understand what matters most to patients when they are making decisions about their healthcare. These studies help researchers and healthcare providers gain insights into the trade-offs patients are willing to make between different treatment options. Patient Preference Studies play a pivotal role in modern healthcare by directly engaging patients to elucidate their priorities regarding treatment options and healthcare outcomes. By gathering and analyzing data directly from patients, these studies provide invaluable insights into creating a patient-centered healthcare system that prioritizes personalized, effective, and satisfactory healthcare experiences.

However, such studies are often presented in the form scientific papers and most of these preferences data is codified in tables, figures and charts. Researchers and data analysts may want to collect these patient data for further analysis: this poses a significant challenge for the data extraction process of their workflow due to the unstructured nature of these documents. The standard current way of doing this extraction, is by a manual inspection of the paper. This process may take even several hours just for a single paper, therefore the need for a innovative solution that automates this process or at least aids the users in order to be more time-efficient.

The objective of this paper is to propose the development of automated system capable of efficiently extracting data from Patient Preference Studies scientific papers. This automated system aim to streamline the data extraction process, thereby overcoming the limitations of manual extraction, which is both time-consuming and impractical. The system will then organize this data in a structured format within a database or spreadsheet, ensuring high quality, accuracy, and error-free results. Moreover, this system is designed to adapt to a growing volume of articles and evolving data formats, ensuring scalability and sustainability in handling the increasing number of studies published regularly.

The methodology presented in this paper revolves around the integration of three key models: PDFigCapX[10], ChartOCR[8], and Table Transformer[15]. PDFigCapX is utilized for extracting images from PDF documents, ChartOCR specializes in the data extraction from charts images. Additionally, the Table Transformer model is employed for efficiently extracting tabular data from PDF articles. By combining the capabilities of these models along with many pre- and post-processing intermediate steps, we create a pipeline capable of successfully extract data directly from PDF files and organize it in a cohesive structured output format. Finally, we employ a range of output metrics to rigorously evaluate the performance of these models.

2 RELATED WORK

Unfortunately, no single, automated method exists for reliably extracting data from graphs and tables embedded within PDF documents. This forces researchers to rely on tedious manual extraction, often visually inspecting elements and transcribing values into programs like Excel. Tools like PlotDigitalizer[14] provide some assistance by offering real-time pointer coordinates during graph observation, but their utility remains limited in terms of efficiency and accuracy.

Despite these challenges, the current research landscape offers promising avenues for a potential solution. By strategically combining methods from several related areas, we might achieve the desired data extraction capabilities. These areas include:

- Image and table extraction from PDF documents: Techniques are available to isolate graphical elements like tables and images from the PDF format.
- Data extraction from tables: Methods exist to analyze table structure and extract data from individual cells.
- Data extraction from chart images: Computer vision algorithms can identify key components within graph images (axes, labels, data points) and extract numerical values.

2.1 Detecting images from PDF files

PyPDF2[3] stands as a popular choice within the Python community for PDF manipulation tasks, valued for its user-friendly interface and straightforward functionality. However, it's important to acknowledge that PyPDF2 does have its limitations, especially when handling intricate PDF files or incorporating newer PDF features. Notably, PyPDF2 does not account for image captions and text annotation. This process encounters challenges, particularly when dealing with raster-PDFs where text has been generated through OCR, as deep learning document layout analysis methods trained with newer or vector-based PDFs may not perform as effectively.[11]

On the contrary, PdfigCapx [10] presents a pioneering method specifically designed to extract figures and their corresponding captions from a dataset containing both vector and raster-based PDFs. Leveraging a fusion of deep learning object detection methods and heuristic techniques, PdfigCapx analyzes scans of article pages along with the text features generated from scan processing.

2.2 Extracting data from chart images

While humans can easily interpret the underlying data in charts, machines face challenges. Existing methods for extracting data from chart images fall into two main categories:

- Rule-based methods like ChartText[13] and ChartSense[7] approaches heavily rely on manually engineered features and rules to locate chart elements in the image. These work well when charts follow very specific, predictable structures but lack adaptability to diverse chart styles.
- End-to-end deep learning methods such as Faster R-CNN and YOLO train powerful object detection models and offer high accuracy, but they are limited to the chart types they've been trained on, meaning that each chart type will require its own dedicated network.

ChartOCR[8] is a novel hybrid approach to the problem that combine the strengths of both by first detecting chart keypoints for chart component identification (like axis positions or legend locations) in a way that's independent of style, and then applying rules to connect these keypoints to reconstruct specific chart elements (bars, lines, etc.). This provides a uniform process across chart types, adaptability to new styles, and the ability to generate semantically rich intermediate results akin to rule-based methods.

2.3 Table Detection (TD)

The detection of tables in PDF documents is a critical component in the field of automated document processing. This phase involves identifying and isolating areas of the document containing tabular structures. The main challenges in this area include the variety of PDF document formats and layouts and the need to maintain the structural integrity of the extracted data. Numerous approaches have been proposed to address this challenge. Among these, the use of Object Detection models, such as YOLO (You Only Look Once) [5], has garnered significant attention. These models can automatically detect objects within an image, including the outlines of tables in PDF documents. However, some models may have limitations in correctly identifying tables in documents with complex layouts e.g. Yolov8 could have problems with adjacent tables. In our specific context, we explored several options for table detection, including approaches based on YOLO and other Object Detection techniques. We opted for an alternative model that can more accurately handle the presence of adjacent or overlapping tables, avoiding excessive output generation and improving the overall efficiency of extracting tabular data from PDF documents.

2.4 Table Structure Recognition (TSR)

In the context of recognizing table structures in PDF documents, several approaches have been developed for extracting tabular data. One of these approaches is based on analyzing the pixels of the table image, where an algorithm identifies white spaces between words and draws black strips to isolate the words themselves[4]. This method does not require the use of machine learning techniques but instead relies on image processing and segmentation strategies to identify the table structure. Concurrently, another approach focuses on using object detection algorithms, which have been trained on specific datasets to recognize the main components of tables, such as cells, rows, and columns. This method leverages the ability of object detection algorithms to identify and localize objects within an image, thus effectively detecting table structures in PDF documents.[15]

Both approaches offer unique advantages and challenges in table structure recognition. While the pixel-based approach is computationally efficient and does not require specific training, the object detection-based approach can provide higher accuracy and generalization when trained on diverse and representative datasets.

2.5 OCR

OCR (Optical Character Recognition) is a technology that aims to convert images containing printed or handwritten text into editable digital text. This technology is widely used for text extraction from paper or digital documents, making information otherwise unavailable in digital format accessible and usable. However, OCR faces several challenges when it comes to extracting tabular data from PDF documents. One of the main issues is the complexity of tables, which can feature varied formatting, complex structures, and merged cells. These characteristics can make it difficult for OCR algorithms to correctly identify cell boundaries, leading to errors in data extraction. To address this challenge, various approaches have been proposed by the research community. Some have focused on developing image pre-processing techniques to improve the quality of input for OCR algorithms.

3 METHOD

Figure 2 illustrates a high-level overview of the project's pipeline. Given that the methodologies for handling tables and charts follow

different procedures, we divided the main pipeline into two different branches, using dedicated models and pre-processing steps tailored for each data type. Our strategy involves the integration of three different models, which are going to be explained in more detail in the following sections.

3.1 PDFigCapX

The PDFigCapX model is designed to extract figures and their captions from biomedical documents. The framework comprises six steps, illustrated in Figure 3: PDF parsing, basic layout analysis, detection of captions and figure regions, identification of graphical content, disambiguation of figure regions, and expansion of figure captions.

During PDF parsing, the model parses each document, separating it into text-stripped pages containing graphical content and an HTML file with text content only. The HTML files provide essential layout information, such as column count and content region boundaries. In the second step, the model extracts all layout details from the PDF. Next, in step 3, potential caption headers are detected by scanning the HTML file for lines starting with terms like "Fig" or "FIG," recording their positions within the page layout.

Step 4 involves using the Connected Component Analysis (CCA) [6] tool to identify graphical content that could be potential figure components on each text-stripped page. Step 5 integrates information from previous steps to disambiguate and pinpoint actual figure regions within each PDF. Finally, in the last step, the associated figure caption is constructed by expanding the continuous text block adjacent to the detected potential caption header positions from Step 3.

3.2 ChartOCR

As depicted in Figure 4, the framework is structured into three main components: common information extraction, data range extraction, and type-specific detection.

Common information extraction encompasses key point detection and chart type classification. Key points are defined differently based on the chart type. (for example, in bar charts they represent the top-left and bottom-right corners of each bar). The framework adopts a modified version of CornerNet[9] with Hourglass Net[12] backbone for key point proposal. The key point detection network generates a probability map highlighting key point locations, with three channels indicating top-left, bottom-right, and background positions. The corner pooling layer, borrowed from CornerNet, aids in localizing key points by performing max-pooling in horizontal and vertical directions.

Data range extraction determines the range of numerical values within the plot area. For isolating y-axis labels, it assumes they are positioned exclusively on the left-hand side of the plot area. Once the plot area is identified, an OCR tool along with a Data Range Estimation algorithm is employed to calculate the data range and pixel range for mapping points to actual data values.

Type-specific detection employs rules specific to each chart type to identify data components (e.g., sectors for pie charts). By integrating these data components with the data range, numerical chart values can be obtained effectively.

3.2.1 Limitations and extensions. ChartOCR is not able to generalize across unseen chart types, as it employs hand-crafted rules for each specific chart type. Therefore chart types different than bar charts, pie charts and line charts are not officially supported.

Furthermore, it is not able to extract text data from the x-axis and from the legend natively: it is just able to locate these areas in the image, but further post-processing steps are required to extract also these kind of data. By leveraging OCR tools and clustering algorithms we were able to locate text clusters in the image, and through hand-crafted rules we were able to successfully match each text cluster to the corresponding data points previously extracted.

Finally, as we will show in the Results section, the original Data Range Extraction algorithm has shown poor performance on our dataset, therefore a new custom algorithm was built to improve the extraction accuracy. Further details can be found in the code in our GitHub repo.

3.3 Table Transformer

DETR (DEtection TRansformer)[1] is an innovative deep learning model developed by Facebook AI Research for object detection in images. Unlike traditional approaches, which typically rely on convolutional neural networks combined with region-based detection algorithms, DETR adopts a fully transformer-based architecture, as shown in Figure 5 . Its architecture comprises a convolutional backbone, such as ResNet-50 or ResNet-101, followed by an encoder-decoder Transformer. This unique design enables DETR to directly predict the positions and classes of objects in an image without the need for generating region proposals. This end-to-end trainable model has demonstrated competitive performance in various object detection tasks. The model is trained using the PubTables-1M dataset[15], developed by Microsoft, consisting of a total of 947,642 tables extracted from scientific articles selected from the PMCOA (PubMed Central Open Access) corpus. These tables are annotated to provide detailed spatial information, including bounding boxes for rows, columns, cells and the entire table. The dataset development process includes alignment and completion phases to ensure consistency and data quality. The TD task and the TSR task is modeled as object detection with images as input, thus will be trained one DETR model for TD and one DETR model for TSR.

3.3.1 For Detection. For TD, there are 2 object classes: table and table rotated. The table rotated class corresponds to tables that are rotated counterclockwise 90 degrees. As a preprocessing step, the PDF pages are converted into images using the pdf2image[2] library, a Python (3.7+) module that wraps pdftoppm and pdftocairo to convert PDF to a PIL Image object. This conversion is essential since the DETR model requires image inputs rather than direct PDF documents. Using pdf2image ensures transformation while preserving the visual content of the original PDF pages, enabling effective data extraction from tables by the DETR model.

3.3.2 For Structure Recognition. The model for TSR uses six object classes: table, table column, table row, table column header, table projected row header, and table spanning cell. In the pre-processing stage for this task, the images of the tables are analyzed to detect if they are rotated, based on insights obtained from the output of the previous model. If rotation is detected, the images are rotated to the

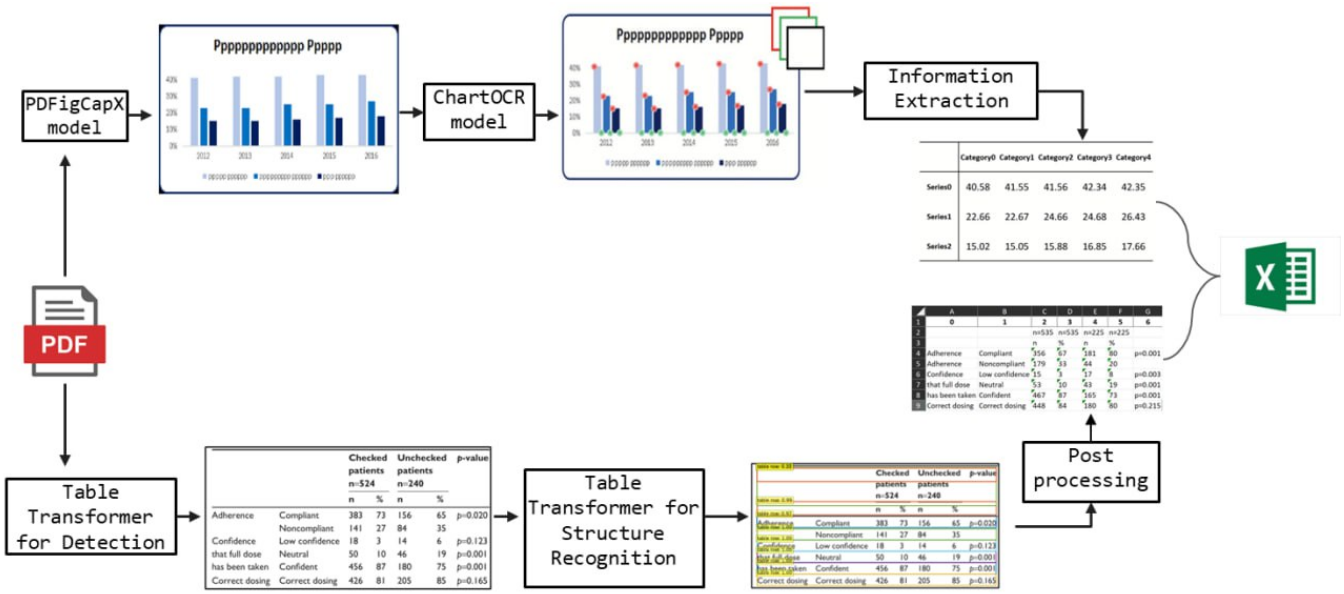

Figure 2: The Overview of System Architecture

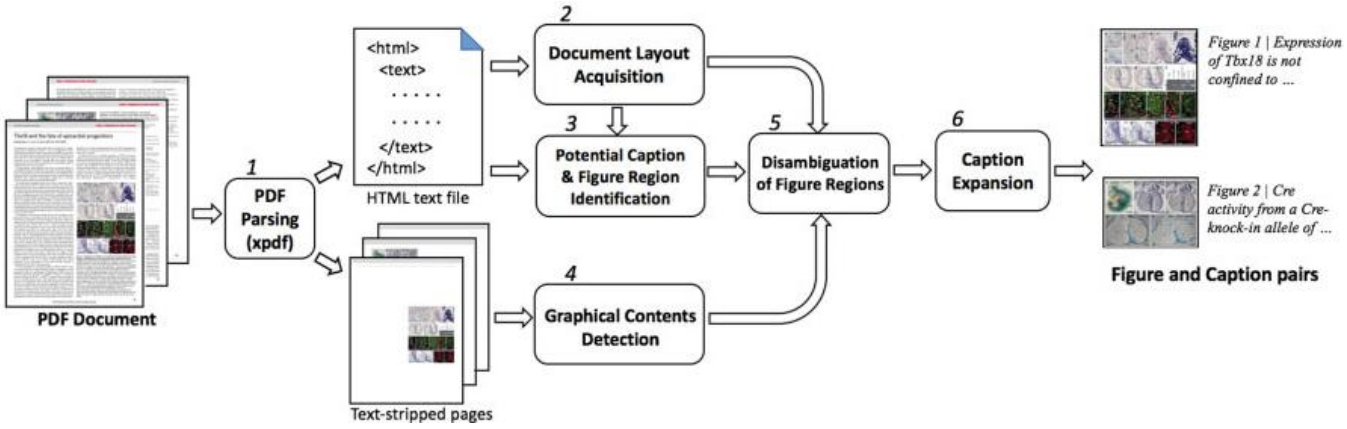

Figure 3: The Processing of PDFigCapX

correct orientation to ensure accurate processing. Additionally, a black border is added around the tables by expanding the image with the same background color. This step helps in enhancing the visual clarity of the table boundaries, facilitating better segmentation and extraction of table.

3.3.3 Post processing . Given that the model output provides the position of cells within tables extracted from PDF documents, we have developed a process to transform this information into a structured Excel file. After identifying the cells, we proceed by splitting the table image into smaller images, each corresponding to a single identified cell. Subsequently, we apply an Optical Character Recognition (OCR) algorithm to each of these cell images to extract the text contained within them. By utilizing both the model output and the OCR output, we obtain detailed information about the words and their positions within the table. This data is then used to construct a dataframe representing the extracted table. Finally, using Python libraries such as Pandas, we convert this dataframe into an Excel file, where each sheet corresponds to a table extracted from the original PDF. This approach allows us ensuring high accuracy during the evaluation and consistency in the final results.

4 EXPERIMENTS

4.1 Dataset

Our dataset consists of 36 Open Access research papers in PDF format, provided by the Department of Clinical and Biological Sciences at the University of Turin. These papers contain a variety of charts and tables, in particular:

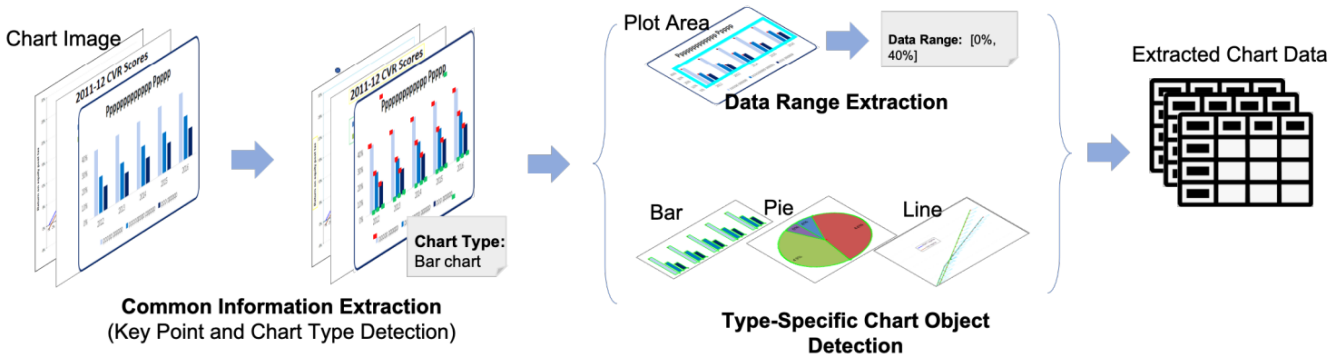

Figure 4: The ChartOCR Model

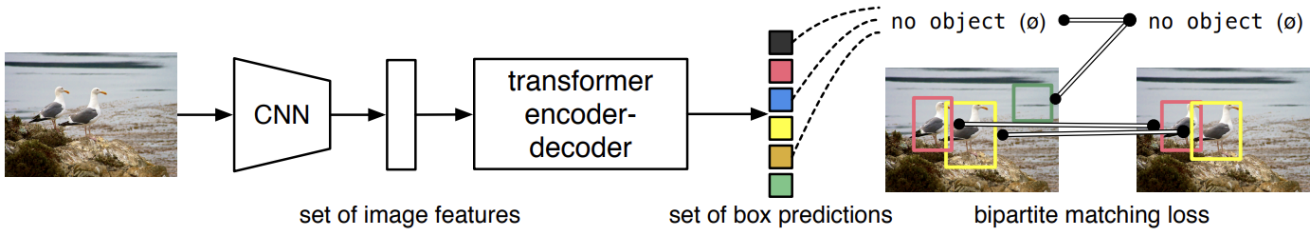

Figure 5: The DETR Model

- A total of 107 charts were identified across all papers. However, only 76 of these graphs were compatible with our model (71% of the total). We decided to focus our analysis on this subset. The incompatible graphs primarily consisted of unsupported types (e.g., box-plots), combinations of multiple types, or complex data visualizations that require subject-domain expert interpretation. While our model may be able to provide partial results for these graphs, we will not explore this in detail in this report.
- A total of 139 tables were identified across all papers. From this, a dataset consisting of 35 tables that are representative and balanced were extracted. Following meticulous review and correction, a labeled dataset was produced. This dataset was specifically prepared to enable the evaluation of the performance of the table data extraction pipeline, ensuring the presence of various types of tables and the information contained within them.

4.2 Table detection Test

During the experiment, different configurations of the model's confidence threshold were examined to assess their effectiveness in identifying tables in PDF documents. Specifically, two configurations were tested: the use of no confidence threshold and the application of a confidence threshold greater than 0.96. The experiment was conducted using all 36 Open Access research papers.

4.2.1 Result. The results obtained showed a significant improvement in table identification when the 0.96 threshold was applied compared to using no threshold. To verify the effectiveness of this detection,the mean of precision, recall, and F-score metrics were calculated. The results of these measurements are reported in the Table 1. These results confirm that the use of a confidence threshold of 0.96 led to more accurate table identification, as evidenced by the minimal variation in recall and the improved values of precision and F-score.

Metric	No Threshold	0.96 Threshold
Precision	0.72	**0.95**
Recall	**1.00**	0.99
F-score	0.83	**0.97**

Table 1: Comparison of table detection performance with and without threshold

4.3 Pre-processing and selection of threshold confidence in TSR

We examined the impact of different image pre-processing techniques and variations in model confidence parameters on the accuracy of structure recognition task. The experimental configurations included two different settings of image pre-processing: one incorporating image expansion with the same background color and adding a black border around the table, and another that did not include these operations. Additionally, we varied the confidence value of the model to examine how it influenced the accuracy of cell box extraction. This experiment was conducted using a representative subset of the dataset of the tables.

4.3.1 Metric for rows and columns. To evaluate the accuracy of table prediction outputs, a specific metric has been developed, focusing solely on comparing the number of rows and columns in the predicted table with those in the correct table. This choice aims to highlight the crucial role of pre-processing and confidence value selection in influencing prediction quality. The metric is calculated as the absolute difference between the number of rows (or columns) in the predicted table and the correct number of rows (or columns), divided by the number of rows (or columns) in the correct table. Then we calculate the mean of these values. In practice, a value closer to zero indicates a more accurate prediction, suggesting less discrepancy between the predicted and correct number of rows (or columns). Thus, this metric provides a clear indication of the model's adaptation to the data, allowing assessment of the precision in predicting the number of rows and columns in tables, crucial for the proper functioning of data extraction and analysis processes.

4.3.2 Results. The results demonstrated that adding pre-processing such as image expansion and adding a black border improved the accuracy in detecting table cells. Although one might assume that increasing the threshold confidence could lead to a reduction in the number of extracted boxes, there is a decrease in the metric, indicating an improvement in performance as the threshold increases. It is discovered, in fact, that the best configuration involves pre-processing and selecting a threshold of 0.8. These findings are summarized in Table 2.

Pre-processing	Confidence	Row Error	Column Error
No Pre-processing	0.2	0.15	0.005
No Pre-processing	0.55	0.11	0
No Pre-processing	0.8	0.12	0
Expansion + Black Border Expansion	0.2	0.11	0.0025
+ Black Border Expansion	0.55	0.0623	0.0025
+ Black Border	0.8	**0.0539**	**0**

Table 2: Selection of pre-processing and threshold confidence.

4.4 OCR and Table accuracy

During the evaluation of the entire pipeline, particular attention was given to the choice of OCR to use. Comparative tests were conducted using two different OCR systems available, Pytesseract and EasyOCR. This approach was adopted to determine which OCR provided the best results in terms of text extraction from PDF documents. The evaluation of the pipeline was performed using a custom metric, which takes into account the accuracy in identifying tables, the completeness in extracting text within the tables, and other relevant factors. This experiment was conducted using the manually labeled dataset specifically created to evaluate the accuracy of the pipeline.

4.4.1 Metric. The proposed method for comparing two tables starts by examining each column of the first table and comparing it with all columns of the second table. This comparison relies on the column similarity metric, where the pair of columns with the highest similarity is sought. Once this optimal pair is identified, the corresponding columns are removed from the pool to prevent re-matching. The overall similarity between the two tables is then calculated by averaging all the column similarities identified. To determine the similarity between columns, the Levenshtein[16] algorithm is used, which compares each element of the columns in search of the maximum similarity. This comparison occurs iteratively, considering all possible combinations of elements between the two columns. A pair is formed only if the Levenshtein similarity exceeds a predefined threshold, in this case, 0.65. At the end of this process, the formed pairs are evaluated, and the F-score is calculated, representing the similarity between the two columns.

4.4.2 Results. The results of this evaluation, summarized in Table 3, showed significant differences between Pytesseract and EasyOCR, with Pytesseract obtaining higher scores according to the custom metric. These results underscored the importance of accurate OCR selection in designing an effective pipeline for automated processing of PDF documents.

OCR	Custom Metric
Pytesseract	**0.776**
EasyOCR	0.729

Table 3: Results of pipeline evaluation with Pytesseract and EasyOCR using Custom Metric.

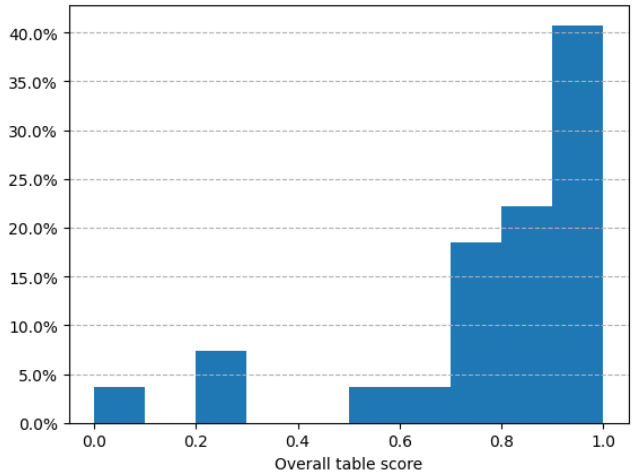

Figure 6: Distribution of table overall scores. (Pre-processing + Pytesseract)

4.5 Chart detection Test

The evaluation of the PDFigCapX model's performance on our 36 PDF papers is shown in Table 4. A recall of 0.84 shows good performance in extracting most charts present in papers. The main problem found was that in many papers two or more charts can

be presented as a single figure/image, and the model struggle in these cases as it may recognize only a single one of them or the whole group together, invalidating the subsequent step of chart extraction. Overcoming this limitation could greatly improve the accuracy, however outside of this edge cases the performance is satisfactory.

Metric	PDFigCapX
Precision	0.76
Recall	0.84
F-score	0.82

Table 4: Accuracy of Chart Detection

4.6 Chart extraction evaluation

Chart extraction encompasses four main subtasks: chart component bounding box extraction, data range detection, X-axis label extraction and legend extraction (if present). In the following section we will evaluate each of these individually, by comparing the performance of our method to the default version of chartOCR.

4.6.1 Metrics.

- Chart component: we want to measure whether the chartOCR model correctly detected the data points from the image (still normalized to values between 0 and 1). To do this, we will consider a data point correctly detected if it deviates from the GT (Ground Truth) by a difference less than 1%. Finally, we calculate the number of correctly detected data points over the number of GT data points.
- Data range: extracting the correct data range is critical because if it is not done, all numerical data will be wrong since they will follow a different scale. Therefore, we will consider the data range pair (dr_{min}, dr_{max}) correct only if both numbers match the GT.
- Legend: the data extracted from the legend consists of a sequence of labels (strings). We choose to use the Levenshtein metric for string similarity and consider them "similar" if the value is greater than 0.7. Each label is compared with the label in the corresponding position of the GT legend. The final value of the legend will be the number of correctly predicted labels divided by the number of GT labels.
- x-axis labels: the data extracted from the x-axis consists of a sequence of labels (strings). We use the same metric as the legend.
- Overall: finally, we construct a metric to evaluate the overall chart extraction accuracy by taking into consideration all the previous elements in a weighted average formula, giving the greatest importance (weight) to the accuracy of numerical data.

$$ACC_{overall} = 0.7 \cdot DR \cdot ACC_{chartcomponents} + 0.15 \cdot ACC_{legend} + 0.15 \cdot ACC_{axislabel}$$

where DR = 1 if data range is correct, else DR=0.

Metric	Baseline	Our method
Chart components	**0.80**	**0.80**
Data range	0.59	**0.95**
x-axis labels	0	**0.79**
Legend	0	**0.59**
Overall	0.47	**0.76**

Table 5: Performance of chart extraction subtasks

4.6.2 Results. From the results shown in Table 4, we can draw some interesting observations. As already anticipated, the performance of the chart component detection is already satisfactory in the original version of chartOCR: we therefore decided to leave this part unchanged, avoiding fine-tuning with additional data. Regarding the data range extraction, the performance of the original algorithm is insufficient: this is the reason why we decided to introduce a new algorithm built by us, greatly increasing the accuracy. Finally, we measure the performance of the new features introduced not present in the original chartOCR, namely the extraction of labels from the x-axis and legend, to give an idea of the accuracy of our methods. The values show a good but not ideal performance, demonstrating that there is still room for improvement in both the algorithm and in the third-party OCR tool.

4.7 Overall considerations on performance

Figure 5 and figure 6 show respectively the distribution of overall scores among tables and charts. About 82% of tables obtained a score greater than 0.70 and about 79% of charts obtained a score greater than 0.70. We deem this statistics important as the value of 0.70 can be considered as the threshold that separates a successfully accurate extraction from an insufficient one. A score greater than 0.70 means (from our subjective observations in qualitative extraction samples) that if the user is interested in manually fixing the output prediction in order to achieve a 100% perfect data quality, the amount of manual

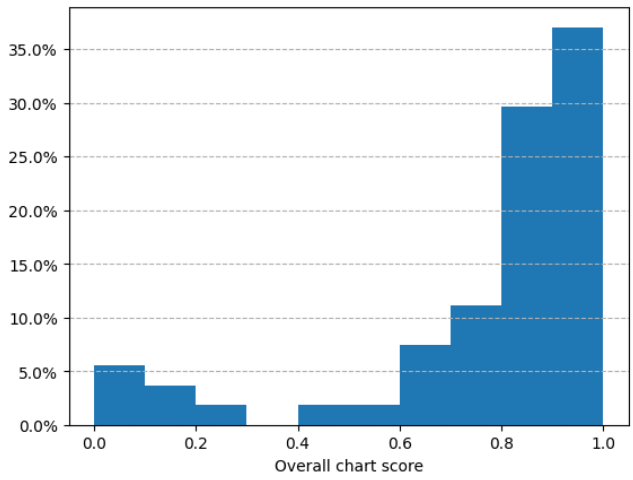

Figure 7: Distribution of chart overall scores.

work needed is minimal, while lower scores may signify a partial or wrong extraction therefore requiring more manual work.

5 CONCLUSION

5.1 Final outcome

In this section, we discuss the tangible outcomes of the project and the added value it brings compared to the situation before the introduction of our solution.

- About 70% of data encoded in a paper is now automatically extractable: considering the overall results and statistics reported in the previous section, we estimate that on average, for a PPS paper, about 80% of the data related to tables can be automatically extracted with a high value of accuracy, while for graphs this value drops to 50%. This value is due to the fact that despite the extraction model obtaining an excellent performance, the great variety of complex and unique types of graphs makes the model ineffective for a good part of the input data, making a good part of the data encoded in the papers inextractable with automatic methods. Furthermore, the detection model still has room for improvement due to the problem of multiple graphs in a single figure. However, considering the challenging nature of chart data extraction and that the starting point was basically zero, we consider this number as a success. Overall, considering that tables are more common than graphs, we estimate that about 70% of the data encoded in the papers is successfully extracted automatically.
- For a single PPS paper, time required for extracting all encoded data is reduced by at least 70%. If extracting everything from a paper without missing a single table or chart is important, our system can be used along manual methods to greatly reduce the time required, either by using conventional manual extraction methods or by performing some extra data cleaning manual operations before feeding data to the system to further increase the extraction accuracy. For example, a paper that would have taken an hour to extract now would require approximately just 15 minutes.
- The system effortlessly scales from individual papers to vast collections: researchers can now confidently process hundreds or thousands of papers, unlocking the power of big data approaches within their scientific field. This has the potental to change research practices as a whole: meta-analyses that were previously considered impossible or too laborious are now enabled, more sources can now be incorporated in research studies, data collection for advanced analytics projects is much more accessible than before.

5.2 Summary

This project successfully developed an automated system for extracting data from Patient Preference Studies (PPS) scientific papers. The system significantly reduces the time and effort required compared to manual extraction, allowing researchers to analyze larger quantities of PPS data. Our approach successfully combines three key models: PDFigCapX for image detection, ChartOCR for chart data extraction, and the Table Transformer for handling tabular data. The system effectively extracts approximately 70% of the encoded data within PPS papers and reduces the extraction time of a single paper by at over 70% compared to a fully manual operation. This automation empowers researchers to dedicate more time to more value-adding activities, saves costs of manual extraction time, and enables the use of large-scale datasets for conducting research studies, potentially transforming research practices within the field of patient preference studies.

5.3 Future work

To further enhance the capabilities and impact of this system, several promising avenues for future work exist. First, refining and extending the data extraction model would address its limitations in handling complex and diverse chart types. This could involve incorporating domain-specific knowledge and image processing techniques, as well as utilizing large labeled datasets for model fine-tuning. Secondly, there is room for improvement in the chart detection step, as it struggles when more than one chart are blended in a single figure of the paper. Thirdly, our system's performance is heavily dependent on third-party OCR services: by simply updating it to a more powerful one we may witness an increase in accuracy without even changing anything else in the system's architecture. Finally, a user-friendly interface with visual output and options for manual adjustments would empower researchers without extensive technical expertise to interact with the system.

REFERENCES

[1] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. *CoRR*, abs/2005.12872, 2020.

[2] Edouard Belval. pdf2image. https://pypi.org/project/pdf2image/.

[3] Mathieu Fenniak. PyPDF2: A Python library to work with PDF files. https://pypi.org/project/PyPDF2/, 2022. Accessed: 23 February 2024.

[4] Pascal Fischer, Alen Smajic, Alexander Mehler, and Giuseppe Abrami. Multi-type-td-tsr - extracting tables from document images using a multi-stage pipeline for table detection and table structure recognition: from OCR to structured table representations. *CoRR*, abs/2105.11021, 2021.

[5] Foduu.com. Table detection and extraction. https://huggingface.co/foduucom/table-detection-and-extraction, Year of publication.

[6] Rafael Gonzalez and Zahraa Faisal. *Digital Image Processing Second Edition.* 06 2019.

[7] Daekyoung Jung1 Wonjae Kim1 Hyunjoo Song1 Jeong in Hwang1 Bongshin Lee2 Bohyoung Kim3 Jinwook Seo1. Chartsense: Interactive data extraction from chart images.

[8] Jinpeng Wang3 Junyu Luo*1, Zekun Li2 and Chin-Yew Lin3. Chartocr: Data extraction from charts images via a deep hybrid framework.

[9] Hei Law and Jia Deng. Cornernet: Detecting objects as paired keypoints. *CoRR*, abs/1808.01244, 2018.

[10] Pengyuan Li, Xiangying Jiang, and Hagit Shatkay. Extracting figures and captions from scientific publications. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, CIKM '18, page 1595–1598, New York, NY, USA, 2018. Association for Computing Machinery.

[11] Jill P. Naiman. Generalizability in document layout analysis for scientific article figure caption extraction, 2023.

[12] Alejandro Newell, Kaiyu Yang, and Jia Deng. Stacked hourglass networks for human pose estimation. *CoRR*, abs/1603.06937, 2016.

[13] Joao Pinheiro and Jorge Poco. Charttext: Linking text with charts in documents. *CoRR*, abs/2201.05043, 2022.

[14] Ankit Rohatgi. Webplotdigitizer. https://automeris.io/WebPlotDigitizer/.

[15] Brandon Smock, Rohith Pesala, and Robin Abraham. Pubtables-1m: Towards comprehensive table extraction from unstructured documents. *CoRR*, abs/2110.00061, 2021.

[16] Wikipedia. Levenshtein distance, 23 December 2023, at 14:23 (UTC).