Optimizing Text Extraction From Complex Newspapers

Ishan Mehta
Department of Computer Science
Stanford University
ishanm@stanford.edu

Ohm Patel
Department of Computer Science
Stanford University
ohmpatel@stanford.edu

Abstract

This project aims to perform accurate text extraction on century-old newspaper clippings utilizing tools such as GPT-4o and Tesseract. Due to the agentic nature of the course, we aimed to create a pipeline which made calls to various agents, preserving autonomy and enhancing the accuracy of our solution. We began by running four baselines, two with Tesseract and two with GPT-4o, in which we found the importance of accurate segmentation of text. Both tools were very accurate at text extraction when given a cropped image, hitting F1 scores over 0.89. For our pipeline, we use calls to both of these tools with Tesseract primarily handing the segmentation and GPT-4o more focused on the extraction and validation of the text. On average, we were able to detect about 73 percent of all articles in a given newspaper, and when we condition our results on this number, achieved an F1 score around 0.6. This paper breaks down some of the other work done in this field, outlines the tools we used in more detail, provides details on our exact methods, and lists our results and analysis on the project.

1 Introduction

The notion of text extraction has been at the forefront of many cutting-edge research papers, specifically via tools like GPT-4o, Tesseract, and more. While text extraction on small subsets of text is improving, there is very little work done on extracting from larger and more complex pieces of text. In this project, we focus on newspapers from the 20th century, most of which are filled with older language, fonts, images, and formatting.

One of the largest problems we faced in this task was the lack of fully verified truth text to compare our extraction results against. There was text that came within our dataset, but it included many illegible words, random characters, and sometimes entirely wrong translations of the newspaper text. For this reason, we not only focused on the accurate segmentation and extraction of newspaper text, but also in finding mechanisms to evaluate our work without relying entirely on the given truth text.

In order to tackle such a daunting task, we employ an agentic solution, splitting our solution into three parts: segmentation, extraction, and validation. NVIDIA defines agentic Artificial Intelligence as a system which uses sophisticated reasoning and iterative planning to autonomously solve complex, multi-step problems. In our use case, we utilize agents in all three parts of our process, helping to create an entirely automated and context-driven pipeline.

Segmentation refers to the portion of the pipeline in which we create bounding boxes within an image to allow future steps to analyze smaller pieces of text. Extraction is the step of calling models to pull text from the images, including the refinement and double-checking of the extracted text. Finally, evaluation is the measurement of how accurate the extracted text is when compared to the true text.

State-of-the-art vision (SotA) models have been well trained for extraction tasks. However, on their own, they struggle to combine segmentation and extraction texts on complex documents. Focusing specifically on scanned historical complex documents like newspapers, we attempt to merge SotA tools to build a full pipeline for extracting text from such documents. We aim to utilize vision models and LLMs as part of a larger agentic system, to overcome their shortcomings in this task

2 Related Works

The problem of extracting and processing text from complex historical documents has been a significant area of interest in computer vision and natural language processing. Historical documents often pose unique challenges due to their degraded quality, complex layouts, and presence of non-standard fonts or handwritten annotations. Optical Character Recognition (OCR) technologies such as Tesseract have been widely used for text extraction from scanned documents. Tesseract, initially developed by HP and later maintained as an open-source project, has been extensively employed for its ability to handle multi-lingual text and its adaptability to noisy inputs (Smith, 2007). Despite its robustness, OCR systems often struggle with irregular document layouts and text embedded in images, which are common in historical newspapers.

Modern deep learning frameworks like Detectron2 have also been employed for document layout detection. Detectron2, an object detection library developed by Facebook AI, is highly modular and supports state-of-the-art models like Mask R-CNN for precise layout element identification. Its flexibility has made it a popular choice for research involving document segmentation. Other domain specific tools, such as DocBedNet, are examples of applying deep learning frameworks on detection models to segment complex historical documents (Zhu et al., 2022).

Transformer-based models like OpenAI's GPT series have revolutionized text processing by enabling contextual understanding and generation of coherent text. Recent work has explored combining OCR outputs with GPT models to correct errors and enhance text extraction accuracy in historical documents. This hybrid approach effectively handles degraded and irregular text, leveraging GPT's pre-trained knowledge for contextual error correction (Brown et al., 2020).

While significant progress has been made, challenges remain in handling low-contrast images, marginalia, and degraded text. Research in the field is increasingly focusing on multi-modal approaches, integrating visual and textual signals to improve extraction quality. For example, recent studies propose the use of vision-language models like CLIP to jointly understand text and layout information in historical documents (Radford et al., 2021). Moreover, the application of generative AI models in document enhancement, such as noise removal and layout reconstruction, presents a promising direction for future research. Techniques like diffusion models could play a role in generating high-quality document reconstructions for improved downstream text extraction.

3 Relevant Technologies

3.1 Image Segmentation

3.1.1 Tesseract

Tesseract is an Optimal Character Recognition (OCR) tool originally developed at Hewlett-Packard between 1985 and 1994. It is used in the backend of OpenAI and Gemini for their OCR-related queries and has become a leader in the industry. Tesseract offers many services, but we focused on a few features specifically: creating bounding boxes based on an image, extracting text from those bounding boxes, and returning the coordinates of the given bounding boxes. This allows us to have the necessary inputs for future steps in the pipeline including refinement, extraction, and evaluation.

3.1.2 Layout Parser

Layout Parser is a deep learning tool specifically used for detecting layouts in document images. It has fine-tuning and training capabilities as well which can curate segmentation to specific formatting, fonts, and styles. Layout Parser offers a variety of models including PubLayNet, PRImA, Detectron2, and more.

We were interested in Detectron2 as its capabilities closely resembled our segmentation task. However, attempting to use the pretrained model did not seem to do anywhere near as well as Tesseract. This might be due to the complexity of document layout and subpar quality of historical document scans. Another potentially impactful approach is training Detectron2 on a class of complex documents, but that requires intensive compute resources.

3.2 Text Extraction

3.2.1 Tesseract

A direct pipeline from the segmentation done using Tesseract is using Tesseract to get an initial extraction given the bounding boxes it derived. Using the same OCR technology, it extracts text from cropped portions of scanned images. This is beneficial as it provides an initial prediction on the text and lets us incorporate the more advanced GPT models for validation and refinement.

3.2.2 GPT-4o

OpenAI ChatGPT-4o, an open-source model with image input capabilities, was utilized for all stages of the process due to its advanced functionality. For segmentation, we made API calls with queries explicitly designed to split images into article chunks and return the coordinates of the corresponding bounding boxes. During the extraction phase, cropped images at the specified coordinates were inputted, and ChatGPT-4o was tasked with extracting the text from each image. Finally, in the refinement and validation phase, the extracted text was processed using ChatGPT-4o to produce cleaner English while preserving the semantic meaning and sentence structure of the input. This cleaning step significantly improved similarity, BLEU, and F1 scores during the evaluation stage.

4 Approach and Experiments

4.1 Datasets

We utilized two datasets for the testing of baselines and testing of full extraction pipeline. In order to test baselines, we needed to compare sections of articles with accurate bounding boxes. Using the dataset created by Dell Research at Harvard, American Stories (Dell et al., 2023), we had access to JSON files that included downloadable images, bounding information, and true text corresponding to segments. In order to test the pipeline, we use a combination of images from American Stories and Library of Congress's Newspaper Collections dataset (of Congress).

Figure 1: Dataset Example

4.2 Baselines Overview

We employed four baselines for this project in order to gain a better understanding of how current state of the art models perform. This included GPT-4o without any modifications, GPT-4o with segmented text, Tesseract without any modifications, and Tesseract with segmented text.

4.2.1 GPT-4o (No Modification)

GPT-4o with no modifications refers to passing in the newspaper image directly into the model with the prompt, "please extract all of the text from this image and format it into articles." Without any specific prompting to create bounding boxes or further information about the size, layout, or format of the newspaper image, GPT-4o still performed well in fully extracting relevant text. However, we looked to add modifications to get a more accurate starting point.

4.2.2 GPT-4o (With Modification)

GPT-4o with modifications means utilizing the bounding box coordinates from the dataset to crop only the relevant portion of the original newspaper image. Then, text would be extracted and evaluated against the refined version of the dataset's truth text for that bounding box. This approach attempts to evalaute whether GPT-4o has adequate text extraction capabilities if the given image is both smaller and clearer than an entire newspaper page. As shown in our results, this performed better than the original GPT baseline and was used as a major step in our pipeline.

4.2.3 Tesseract (No Modification)

This baseline involves inputting the entire image into a Tesseract API call to retrieve the entire newspaper page text. It did much worse than either GPT baseline and we realized we needed to modify the bounding boxes in order to extract value from a Tesseract baseline.

4.2.4 Tesseract (With Modification)

The modified Tesseract baseline passes in the dataset bounding boxes (same as GPT modified baseline) into a Tesseract API call. It seems to perform slightly worse than GPT-4o, although minor differences can also be attributed to the evalaution methods and refinement to the truth text. This is especially likely because GPT-4o utilizes Tesseract behind-the-scenes for text extraction.

4.3 Our Pipeline

As explained earlier in the paper, we created an entirely autonomous pipeline utilizing agents to handle segmentation, extraction, refinement, validation, and evaluation of newspaper text extraction.

4.3.1 Segmentation

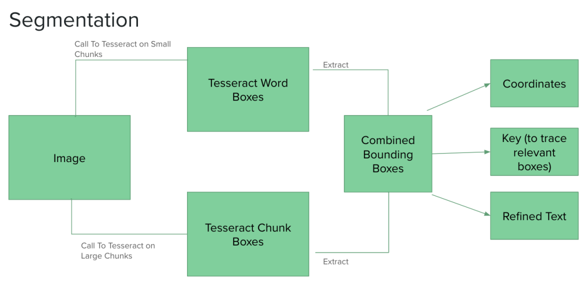

Figure 2: Segmentation Methodology

As demonstrated in the above figure, the segmentation portion of the pipeline begins with a JPEG image. We then use Tesseract to extract both word-by-word and article-level bounding boxes. To achieve the article-level boxes, we start by taking the word-by-word boxes and then concatenate based on what seems to be in the same article. The eventual goal is to get accurate article-level bounding boxes but simply extracting those from Tesseract can lead to missed words or chunks. Once these boxes are achieved, we return them along with their coordinates, keys, and the extracted text from each one.

Through our experiments, we noticed a large variance in both the number and quality of extracted bounding boxes. Specifically, there are many overlapping boxes in which the biggest encompasses the entire image. For this reason, we wanted to prioritize boxes that roughly reflect the size of the article chunks themselves. We do this by getting the average size of bounding boxes

Define the areas of the boxes as:

$$\text{box\_areas} = [(x_2 - x_1) \cdot (y_2 - y_1) \mid (x_1, y_1, x_2, y_2) \in \text{outer\_boxes}]$$

Set the minimum area threshold as:

$$\text{min\_area\_threshold} = \begin{cases} 0.2 \cdot \text{mean(box\_areas)}, & \text{if box\_areas} \neq \emptyset \\ 0, & \text{otherwise} \end{cases}$$

Filter the boxes based on the area:

$$\text{filtered\_boxes} = \{\text{box} \in \text{outer\_boxes} \mid (x_2 - x_1) \cdot (y_2 - y_1) > \text{min\_area\_threshold}\}$$

4.3.2 Extraction/Validation

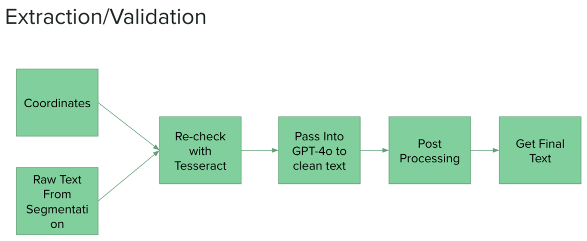

Figure 3: Extraction/Validation Methodology

The extraction and validation process is a critical phase that refines the text and coordinates obtained from segmentation, producing a finalized block of text ready for evaluation. This process consists of three primary steps: Tesseract validation, GPT-4o cleaning, and postprocessing.

The first step, Tesseract validation, involves inputting the extracted text alongside the corresponding cropped image to generate a refined version of the text. While substantial changes are not typically expected at this stage, the second-pass validation of a bounding box often uncovers additional words that were initially overlooked. This step provides a baseline improvement in text extraction accuracy by consolidating the original and newly identified text elements.

The second step leverages GPT-4o to clean and refine the text returned from Tesseract. Tesseract outputs frequently include issues such as irregular characters, inconsistent spacing, and truncated words. The GPT-4o refinement step addresses these challenges, ensuring the text is properly formatted while preserving semantic integrity and stylistic consistency. This step is essential for preparing the text for subsequent evaluation and analysis.

The final step in the extraction and validation process is postprocessing. During this phase, a series of prewritten checks is applied to the text to address residual issues such as spacing errors, improper formatting, and paragraph segmentation.

This step further ensures the text meets quality standards for evaluation. To enhance the reliability of evaluation (discussed in the next section), the GPT-4o refinement and postprocessing steps are also applied to the ground truth text, ensuring consistency in comparison.

4.3.3 Evaluation

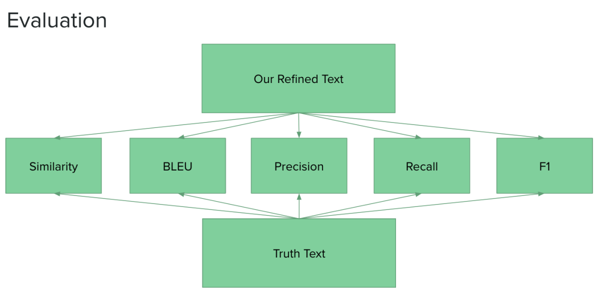

Figure 4: Evaluation Methodology

Due to the goal of wanting exact text extraction, our evaluation methods are focused around finding perfect text matches including BLEU score and cosine similarity. However, there were many examples in which our extracted text was semantically similar or even identical to the truth text, although there were differences in the actual phrases. This leads to lower evaluation scores, but still allows the user to understand what is being said in the newspaper.

We made a very conscious decision to not include evaluation methods such as Word2Vec or SBERT (tools that capture semantic similarity between two pieces of text) because the goal of the task was using vision to extract exact text and not similar text. If this pipeline was used with the goal of simply aiding humans to understand the newspaper content, these types of evaluation mechanisms may be more suited to the specific case.

4.4 Results

Model	BLEU	Precision	Recall	F1
Tesseract	< 0.5	< 0.5	< 0.5	< 0.5
Tesseract + Refining	0.77	0.89	0.90	0.89
GPT-4o	0.72	0.89	0.86	0.87
GPT-4o + Refining	0.79	0.92	0.91	0.91
Full Pipeline	0.79	0.92	0.91	0.91

Table 1: Performance metrics of various models and approaches.

In order to test the segmentation accuracy of the pipeline, we tested on 25 scans of historical documents. Using human review, we achieved roughly 73% accuracy in terms of articles extracted / total articles in the test set. We defined articles extracted as a bounding box covering a majority of an article.

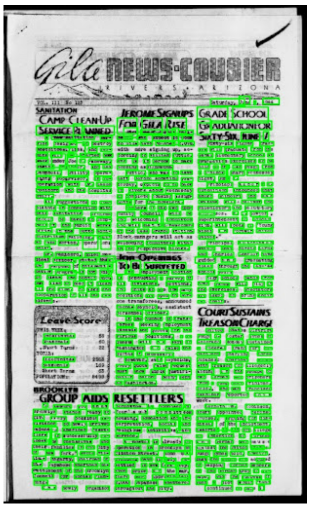

Figure 5: Word By Word Bounding Boxes

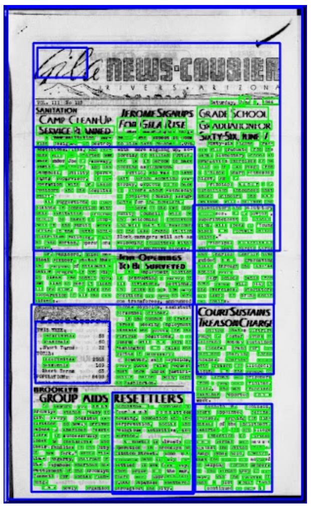

Figure 6: Article-Level Bounding Boxes

5 Analysis

5.1 Refinement Step Benefits

As a part of baseline testing, we compared extraction using Tesseract and GPT-4o with and without text refinement. We had the hypothesis that if there are some flaws in pure text extraction, such as characters out of place or incorrectly formatted sentences, we could resolve them using natural language models as a method of error correcting. We noticed that using refining by making GPT API calls increased scores across all evaluation metrics for Tesseract and GPT-4o.

5.2 Importance of Bounding Boxes

Without even analyzing the pipeline data, it becomes clear from the baselines that accurate segmentation is essential for current text extraction tools to operate effectively. Within our pipeline, when we inputted articles with complex formats, including those with images, smaller articles, or uneven spacing, the segmentation drastically faltered.

5.3 Semantic Similarity

Due to the goal of wanting exact text extraction, our evaluation methods are focused around finding perfect text matches including BLEU score and cosine similarity. However, there were many examples in which our extracted text was semantically similar or even identical to the truth text, although there were differences in the actual phrases. This leads to lower evaluation scores, but still allows the user to understand what is being said in the newspaper.

5.4 Images

Articles containing large, centrally positioned images significantly hindered the ability of Tesseract and GPT-4o to accurately define bounding boxes. This issue arises because these tools often delineate regions based on the bounding box of an image, excluding any text present within that region from recognition. To address this limitation, future work could involve training these models to handle images more effectively or focusing on datasets from newspapers that lack prominent central images.

6 Conclusion

6.1 Summary

In this study, we employed a dataset of historical newspapers from the 20th century to develop a system capable of extracting all text from a given page. Recognizing the availability of advanced vision-enabled models such as GPT-4o and Tesseract, we established and executed comprehensive baselines to assess their performance and limitations. Building on these insights, we designed an autonomous agentic pipeline that integrates tools including GPT-4o, Tesseract, Layout Parser, and others to achieve accurate text extraction from complex layouts. The extracted text was rigorously evaluated using exact-match metrics, including BLEU, precision, recall, and F1 score, to ensure robust performance and reliability.

6.2 Future Work

With increased computational resources, future work could explore training advanced layout parsing models, such as Detectron2, on historical documents to improve segmentation accuracy. Fine-tuning these models specifically for the unique layouts and features of historical newspapers could significantly enhance the quality of text extraction. Additionally, leveraging large language models like GPT-4o or similar systems for more frequent and targeted text extraction and refinement could further improve the accuracy and coherence of the output.

One key challenge faced during the project was the lack of reliable ground truth data for the text, which complicated the evaluation process. Investing additional time and effort into cleaning and verifying the ground truth text would provide a more robust foundation for evaluation, leading to more accurate and meaningful results. Beyond these major advancements, incremental optimizations such as refining bounding box detection, incorporating additional text refinement steps, and improving the smoothness of post-processing pipelines could also contribute to better overall performance. These enhancements collectively would address current limitations and push the boundaries of what is achievable in automated text extraction from historical documents.

7 Contributions

7.1 Ishan Mehta

- Wrote all Tesseract baselines
- Text refinement with Tesseract
- Wrote bounding box extraction algorithm for segmentation
- Sourced and processed dataset

7.2 Ohm Patel

- Wrote all GPT-4o baselines
- Text refinement with GPT-4o
- Worked on evaluation methods and writing evlauation code

- Created validation step and double-checking extracted text

7.3 Both Team Members

Both team members contributed equally on amount of code written, making the poster, recording the video, writing the report, uploading and organizing the GitHub repository, and writing/submitting project status updates across the quarter.

Acknowledgments

This document has been adapted by Steven Bethard, Ryan Cotterell and Rui Yan from the instructions for earlier ACL and NAACL proceedings, including those for ACL 2019 by Douwe Kiela and Ivan Vulić, NAACL 2019 by Stephanie Lukin and Alla Roskovskaya, ACL 2018 by Shay Cohen, Kevin Gimpel, and Wei Lu, NAACL 2018 by Margaret Mitchell and Stephanie Lukin, BibTEX suggestions for (NA)ACL 2017/2018 from Jason Eisner, ACL 2017 by Dan Gildea and Min-Yen Kan, NAACL 2017 by Margaret Mitchell, ACL 2012 by Maggie Li and Michael White, ACL 2010 by Jing-Shin Chang and Philipp Koehn, ACL 2008 by Johanna D. Moore, Simone Teufel, James Allan, and Sadaoki Furui, ACL 2005 by Hwee Tou Ng and Kemal Oflazer, ACL 2002 by Eugene Charniak and Dekang Lin, and earlier ACL and EACL formats written by several people, including John Chen, Henry S. Thompson and Donald Walker. Additional elements were taken from the formatting instructions of the *International Joint Conference on Artificial Intelligence* and the *Conference on Computer Vision and Pattern Recognition*.

References

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners.

Melissa Dell, Jacob Carlson, Tom Bryan, Emily Silcock, Abhishek Arora, Zejiang Shen, Luca D'Amico-Wong, Quan Le, Pablo Querubin, and Leander Heldring. 2023. American stories: A large-scale structured text dataset of historical u.s. newspapers. *Preprint*, arXiv:2308.12477.

Library of Congress. Library of congress.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. 2021. Learning transferable visual models from natural language supervision.

R. Smith. 2007. An overview of the tesseract ocr engine. 2:629–633.

Wenzhen Zhu, Negin Sokhandan, Guang Yang, Sujitha Martin, and Suchitra Sathyanarayana. 2022. Docbed: A multi-stage ocr solution for documents with complex layouts.

A Appendix

Github URL: https://github.com/ohm1129/cs224v-final-project