National University of Singapore

ISY5002 PRS Group Report

Document Reader

PRS Group 10

Group Members

DING YI A0295756J

LIU LIHAO A0296992A

LOU SHENGXIN A0397330A

SHI HAOCHENG A0296265R

YANG RUNZHI A0297296H

Submission Date: October 31, 2024

Contents

1 Introduction

In today's digital landscape, organizations across various sectors are increasingly challenged with managing extensive volumes of unstructured text data. From research papers in academia to reports and contracts in business and law, there is a high demand for systems capable of automatically processing and extracting key information from complex documents. Manual document processing is both time-consuming and prone to errors, especially when dealing with large datasets or intricate document layouts.

This project provides a comprehensive approach to document processing and knowledge extraction, integrating advanced machine learning models and optical character recognition (OCR) technology to tackle challenges in layout analysis, text recognition, summarization, and keyword extraction. Designed with a user-friendly interface, this system allows users to seamlessly upload, process, and retrieve insights from documents, making information retrieval faster, more accurate, and contextually rich. Additionally, the system is especially suited for computer science literature, providing targeted support for extracting and analyzing data from technical documents in this field.

1.1 Background

As industries prioritize data-driven decision-making, efficient tools for document analysis and knowledge extraction have become indispensable. Sectors such as academia, business, and law generate large volumes of documents that require rapid and accurate understanding, categorization, and analysis. Traditional document processing methods—often relying on manual data entry and review—are not only labor-intensive but also introduce a higher likelihood of errors, leading to inefficiencies and potential loss of valuable insights. The development of artificial intelligence (AI) and machine learning models offers transformative potential for automating document processing, bringing improved speed and precision to previously manual tasks.

Document processing faces unique challenges due to the diversity of formats (such as PDFs, scanned images, and Word files) and layouts, which may include tables, figures, diagrams, and multi-column text. Additionally, effective summarization and keyword extraction from lengthy documents require sophisticated natural language processing (NLP) techniques, as key information is often scattered across various sections or spans multiple pages. This demand is even more pressing in technical fields such as computer science, where documents frequently include specialized language, diagrams, code snippets, and dense technical content.

In response to these challenges, our project is designed to meet the specific requirements of computer science-related literature, helping users efficiently extract and interpret complex technical information. The system's user-friendly interface further enhances accessibility, allowing users from various backgrounds to easily interact with and benefit from the tool.

1.2 Project Description

This project develops an automated document processing system that provides layout analysis, text recognition, summarization, and keyword extraction, all accessible through a user-friendly interface. This comprehensive approach is particularly suitable for computer science documents, where complex layouts and technical language present additional challenges. Key components

include:

1. Layout Detection:
Using YOLOv10, a high-performance object detection model, the system can accurately detect structural components within documents, such as headings, paragraphs, code blocks, and images. This capability is essential for segmenting and organizing the extracted content effectively, especially for documents that contain technical sections unique to computer science literature.

2. Text Extraction:
Tesseract OCR, a leading optical character recognition tool, is employed to capture text from images and scanned documents. This step is critical for transforming visual data into editable and searchable text, particularly for printed or scanned materials that are not readily accessible in digital form. The OCR process is especially beneficial for handling complex documents with specialized terminology often found in technical fields.

3. Text Summarization:
To condense large volumes of text, BART, a transformer-based model optimized for summarization, is used to generate concise summaries. This enables users to quickly grasp essential information without needing to read the entire document. The summarization capability is tailored to address dense technical content, highlighting core concepts and findings in a concise manner, which is valuable for reviewing lengthy academic papers or research reports.

4. Keyword Extraction:
Leveraging DistilBERT, a fine-tuned NLP model, the system identifies keywords that capture the main ideas of the text. This enhances document indexing and retrieval, making it easier for users to locate relevant information within large collections. In the context of computer science, this keyword extraction is particularly useful for identifying technical terms, concepts, and research areas, such as machine learning, natural language processing, or network security.

5. Real-Time Wikipedia Linkages:
By integrating with the Wikipedia API, the system provides additional context for extracted keywords, linking them to relevant background information. This adds depth to the extracted data, enabling users to access further insights on key topics, including those that may be highly specialized in computer science.

6. User-Friendly Interface:
The system's interface is designed to prioritize ease of use, with straightforward upload and navigation options. This interface enables users to interact with complex document processing features without needing extensive technical knowledge. A simple dashboard allows users to view summaries, extracted keywords, and contextual links from Wikipedia, streamlining the workflow and enhancing accessibility for a wide range of users.

Through these components, the project offers a robust solution that is particularly suited to the analysis of computer science texts, aiding researchers, developers, and professionals in efficiently accessing and understanding complex technical information. The system can accommodate varied document types and adapt to different content requirements, making it a versatile tool for organizations that manage significant amounts of unstructured data. This tailored approach not only streamlines document processing but also enhances the accuracy and accessibility of essential data, supporting informed decision-making and knowledge discovery in the computer

science domain.

2 Problem Description and Market Analysis

The rapid digital transformation across various sectors has intensified the demand for tools that can efficiently process and analyze large volumes of unstructured text data. Fields such as academia, business, law, and finance are particularly reliant on fast and accurate document processing systems, given the extensive amounts of information that must be processed from formats like PDFs, scanned images, and technical reports. Traditional document processing methods, however, are typically manual or semi-automated, making them time-consuming and prone to error. To meet the growing needs of these fields, there is a critical demand for automated systems that can accurately analyze complex document layouts, recognize and extract text, summarize large volumes of content, and identify essential keywords, all while offering context for more informed decision-making.

2.1 Problem Statement

Efficiently extracting information and accessing relevant background knowledge from diverse digital document formats (e.g., PDFs, scanned images, academic papers) presents significant challenges across fields. Current methods largely depend on manual or semi-automated processes, which are limited in their ability to handle complex layouts and specific content requirements. The following are the primary challenges and essential requirements for a robust document processing system:

1. Handling Complex Layouts:
Documents often contain a variety of elements such as headers, paragraphs, tables, images, and code blocks, all requiring precise recognition and segmentation. An effective system must accurately parse these components to enable efficient extraction and organization of the data, making layout handling a key challenge for comprehensive document analysis.

2. High-Quality Summarization and Keyword Extraction:
Generating meaningful summaries and extracting keywords that are semantically relevant helps improve data accessibility, allowing users to quickly locate and interpret essential information. High-quality summarization and keyword extraction are crucial, especially when processing technical documents or research papers where core insights may be scattered across lengthy text.

3. Contextual Knowledge Integration:
Many document processing tasks in research, business, or academia benefit from contextual information linked to keywords or main topics. Existing systems often lack this integration, placing a greater burden on the user. Connecting extracted terms to reliable external sources, like Wikipedia, adds immediate relevance and depth to the information, enhancing the user's understanding.

4. Scalability and Performance Optimization:
For high-demand environments, document processing systems must scale effectively without compromising speed or accuracy. Achieving this balance is crucial to ensure the system can manage multiple documents simultaneously and provide consistent results, making it practical

for real-world applications.

This system, with its automated approach to document handling, could serve a broad user base. For academic researchers and students, it facilitates rapid review of technical literature; for business analysts, it supports quicker data-driven decision-making; and for legal and financial professionals, it provides accurate and efficient processing of complex documents. By reducing manual effort and improving information retrieval accuracy, the system holds potential to significantly streamline document management across sectors.

2.2 Business Model and Solution

The need for efficient document processing solutions is increasing, particularly in academia, business, and law. This system addresses this demand by automating layout analysis, text extraction, summarization, and keyword generation while also integrating contextual knowledge through resources like Wikipedia. By enhancing accessibility to key information, the system supports faster and more informed decision-making.

2.2.1 Target Customer Segments

The system is designed for use by academic institutions, research organizations, government agencies, and professionals in document-intensive fields like finance and law. By automating essential tasks in document processing, the system enables users to focus on higher-level analysis, improving productivity and accuracy.

2.2.2 Revenue Model

To support continuous development and cover operational costs, the system utilizes a subscription-based revenue model with customizable options for academic, corporate, and governmental clients. Additionally, the system offers industry-specific configurations, such as for legal or academic document analysis, which provide additional revenue opportunities. Partnerships with educational institutions offer another revenue stream by providing training programs that introduce students and professionals to advanced document processing techniques.
This diversified revenue model ensures the system remains sustainable while also allowing it to evolve with user needs and maintain relevance in a competitive market.

2.3 Market Research and Competitive Analysis

With the advent of AI-driven models, document processing is shifting away from traditional, template-based approaches towards more adaptable machine learning models. This shift addresses the growing need for systems capable of handling complex, unstructured data. Yet, many existing tools encounter limitations, particularly when processing documents with intricate layouts or specialized text.

2.3.1 Limitations of Existing OCR and Summarization Tools

Tools like Google's Tesseract OCR work well with clean, standardized text but often struggle when faced with multi-column formats, low-resolution images, or documents containing dense tables and graphics. Similarly, modern text summarization and keyword extraction models based on transformer architectures (such as GPT-3 and BERT) have improved accuracy but require extensive domain-specific training to reliably handle technical content. Without this customization, these models may fail to generate coherent and relevant summaries, especially for specialized fields. Likewise, existing keyword extraction tools, including KeyBERT

and TF-IDF-based approaches, lack the capability to capture nuanced information without domain-specific adaptation, and generally do not support linking keywords to external knowledge sources, which limits their utility in providing comprehensive background information.

2.3.2 Advantages of Our Proposed System

Our system addresses these limitations by combining state-of-the-art AI techniques to provide a comprehensive solution for document processing and knowledge extraction. It utilizes YOLOv10 for layout detection, Tesseract OCR for text extraction, BART for summarization, and DistilBERT for keyword extraction, enabling it to handle a wide range of document types and complex layouts. The integration of Wikipedia-based contextual links further enhances the user's experience by providing relevant background information alongside extracted keywords, improving both accessibility and understanding.

2.3.3 Differentiation and Scalability

Designed to be scalable, this system can process high volumes of documents without sacrificing accuracy, making it suitable for environments requiring consistent performance. Unlike traditional document processing tools, it offers precise layout analysis, advanced text extraction, and contextual keyword linking, making it an ideal choice for data-intensive industries like finance and law. The scalable, user-centered design of this system enables it to meet evolving user needs effectively, positioning it as a competitive solution in the rapidly advancing AI-powered document processing market.

3 System Architecture and Modeling

This section outlines the architecture and modeling techniques used in the system, detailing the components and processes that enable efficient document processing and knowledge extraction. The system combines advanced machine learning models with traditional OCR tools, creating a modular architecture that performs layout detection, text recognition, summarization, keyword extraction, and contextual linkage. These components work together in a streamlined pipeline, allowing users to efficiently extract and analyze structured information from complex documents, such as PDFs, scanned images, and academic papers.

3.1 System Architecture

The system architecture is designed with modularity and flexibility in mind, enabling each component to perform specific tasks that contribute to the overall functionality of the document processing workflow. Key components include:

1. Document Layout Detection:
Using the YOLOv10 model, the system first analyzes document images to detect structural elements such as headers, paragraphs, tables, and images. YOLOv10's high-speed detection capabilities make it suitable for identifying varied document layouts, providing a foundation for further processing steps.

2. Text Extraction:
After identifying layout elements, Tesseract OCR is employed to extract text from each identified section. Tesseract's ability to handle multiple languages and custom configurations ensures that it can capture text accurately, even in challenging formats, transforming visual data into

searchable and editable text.

3. Summarization:
Once text has been extracted, the BART model, a transformer-based summarization tool, generates concise summaries of dense content. This component enables users to quickly access essential information without needing to read entire documents, enhancing efficiency for research and review tasks.

4. Keyword Extraction:
The DistilBERT model is utilized to identify relevant keywords within the extracted text. This enables efficient indexing and retrieval of key topics and themes within the document, which is especially useful for identifying technical terms and core ideas in complex academic or professional materials.

5. Contextual Linkage:
To provide additional context for keywords, the system links extracted terms to real-time Wikipedia content via the Wiki API. This integration allows users to access background information quickly, supporting deeper understanding of specialized or unfamiliar topics.

By combining these components, the system provides a comprehensive solution for automated document processing. The architecture is flexible enough to accommodate additional models or tools, allowing for further customization to meet specific industry needs, such as legal document analysis, academic research, or business report generation.

3.2 YOLOv10 for Text Detection

YOLOv10, which stands for "You Only Look Once" version 10, is an advanced object detection model known for its ability to quickly and accurately detect objects in images. Unlike traditional object detection algorithms that use multiple stages, YOLOv10 uses a single-stage operation to localize and classify objects through the network at once. This single-stage approach significantly reduces computation time, making YOLOv10 ideal for applications requiring real-time detection and complex layout analysis(Redmon et al., 2016).

The core mechanism of YOLOv10 relies on dividing the input image into a grid, where each grid cell is responsible for predicting bounding boxes and class probabilities for objects that appear within it. YOLOv10 builds on the foundation of previous YOLO versions but introduces advancements in the network's architecture to enhance speed and accuracy. The model can be described by the following components and equations:

1. Grid-Based Object Detection:
YOLOv10 splits the input image into an $S \times S$ grid, where each cell i within the grid predicts B bounding boxes and C class probabilities. The output for each bounding box includes five values: the center coordinates (x, y), width w, height h, and a confidence score P_{obj}, which indicates the probability of an object being present:

$$P_{\text{obj}} \times \text{IOU}_{\text{pred}}^{\text{truth}}$$

where $\text{IOU}_{\text{pred}}^{\text{truth}}$ represents the intersection over union between the predicted and true bounding boxes.

2. Bounding Box Prediction:
Each grid cell predicts bounding boxes in a manner that prioritizes the detection of larger and more centered objects. YOLOv10 applies anchor boxes—predefined bounding box shapes—to adjust predictions, refining them based on specific document layouts or image characteristics. This anchor box approach is particularly useful for handling objects of different shapes and sizes across varied document formats.

3. Loss Function and Optimization:
YOLOv10's loss function integrates localization, confidence, and classification losses, aiming to optimize object detection accuracy. The total loss L is defined as:

$$
\begin{aligned}
L = & \lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \right] \\
& + \lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[(\sqrt{w_i} - \sqrt{\hat{w}_i})^2 + (\sqrt{h_i} - \sqrt{\hat{h}_i})^2 \right] \\
& + \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} (C_i - \hat{C}_i)^2 \\
& + \lambda_{\text{noobj}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{noobj}} (C_i - \hat{C}_i)^2 \\
& + \sum_{i=0}^{S^2} \mathbb{1}_{i}^{\text{obj}} \sum_{c \in \text{classes}} (p_i(c) - \hat{p}_i(c))^2
\end{aligned}
$$

where $\mathbb{1}_i^{\text{obj}}$ denotes if an object appears in cell i, and $\mathbb{1}_{ij}^{\text{obj}}$ denotes that the j-th bounding box predictor in cell i is "responsible" for that prediction.

4. Non-Maximum Suppression (NMS):
To refine predictions, YOLOv10 uses Non-Maximum Suppression (NMS), an algorithm that removes overlapping bounding boxes to keep only the most probable detections. NMS calculates the Intersection over Union (IoU) between pairs of bounding boxes, keeping boxes with the highest IoU scores while discarding redundant ones. This technique enhances detection precision and reduces false positives, particularly in densely packed document layouts.

3.3 Tesseract OCR for Text Extraction

Tesseract OCR, an open-source optical character recognition engine originally developed by Hewlett-Packard and later improved by Google, is one of the most widely used tools for extracting text from images (Smith, 2007). Designed to convert images containing textual content (e.g., scanned documents, photographs of text) into machine-readable text, Tesseract OCR supports over 100 languages and can handle both standard and complex layouts, including multi-column texts and non-standard fonts.

1. Preprocessing:
The first step involves preprocessing the image to prepare it for accurate text extraction. This stage includes noise removal, thresholding, and binarization. Binarization simplifies the image by converting it to a black-and-white format, where each pixel is classified as either foreground

(text) or background. The binarization process can be represented as:

$$f(x,y) = \begin{cases} 0 & \text{if } I(x,y) < T \\ 1 & \text{if } I(x,y) \geq T \end{cases}$$

where $I(x,y)$ is the pixel intensity at coordinates (x,y) and T is a threshold value.

2. Character Segmentation:
Tesseract OCR then segments the image into individual characters by identifying connected components. This involves grouping neighboring pixels into larger components, such as letters, words, and lines. A vertical and horizontal projection profile is applied to detect spaces between characters and words. In mathematical terms, the projection profile $P(x)$ for vertical segmentation can be defined as:

$$P(x) = \sum_{y=0}^{H} I(x,y)$$

where H is the height of the image and $I(x,y)$ is the binarized pixel value. Peaks and valleys in the projection profile help distinguish between individual characters and words.

3. Feature Extraction and Character Recognition:
Once segmented, Tesseract OCR uses a combination of neural networks and a feature-based approach to recognize characters. The system extracts features such as character outlines, strokes, and intersections. Each character is then compared to a language model that includes known letter forms and linguistic rules for the selected language. The character matching process uses algorithms such as Levenshtein Distance to compare the recognized characters with a dictionary, minimizing error rates in character recognition:

$$D(i,j) = \min\left(D(i-1,j)+1, D(i,j-1)+1, D(i-1,j-1)+c\right), \quad \text{where } c = 0 \text{ or } 1$$

where $D(i,j)$ represents the distance between two characters.

4. Language Modeling and Contextual Analysis:
Tesseract OCR employs a language model to improve accuracy, especially for words that are difficult to distinguish purely based on their visual features. This model uses n-grams (probabilities of word sequences) to provide contextual guidance for character recognition, especially in ambiguous cases. The probability of a word sequence $W = w_1, w_2, \ldots, w_n$ is calculated as:

$$P(W) = P(w_1) \prod_{i=2}^{n} P(w_i | w_{i-1})$$

This language model allows Tesseract to predict words based on context, reducing misinterpretation of similar-looking characters (e.g., "1" vs. "I").

5. Post-Processing and Error Correction:
Finally, Tesseract OCR performs post-processing to refine the recognized text, using techniques like spell-checking, formatting, and error correction. During this stage, the output is refined to improve readability and accuracy, with adjustments made for factors like spacing, capitalization, and punctuation.

These combined steps allow Tesseract OCR to perform high-accuracy text extraction across

varied languages and complex document layouts, making it a versatile and powerful tool for extracting structured text from images.

3.4 BART for Text Summarization

BART (Bidirectional and Auto-Regressive Transformers) is a state-of-the-art sequence-to-sequence model developed by Facebook AI for natural language processing tasks, particularly text summarization, translation, and text generation. BART is designed to handle a wide range of text generation applications by combining the strengths of both bidirectional and autoregressive models. Its architecture and training approach make it particularly well-suited for abstractive summarization tasks, where the model generates a condensed version of the input text in its own words, rather than simply extracting parts of the original text (Lewis, 2019).

BART's functionality is based on an encoder-decoder architecture that utilizes transformers to process and generate text. This setup enables BART to understand the context of entire text sequences while also learning to generate outputs sequentially, a combination that has proven highly effective for producing coherent, context-aware summaries.

1. Pretraining with Text Corruption:
BART is pretrained by corrupting input texts in various ways and then learning to reconstruct the original text, similar to denoising autoencoders. This approach allows the model to learn contextual understanding and develop robust language representations. Some common text corruption methods used in BART's training include token masking, token deletion, sentence permutation, and document rotation. Formally, let x represent the original text, and $\hat{x}$ the corrupted text. The pretraining objective is to minimize the reconstruction loss:

$$L_{\text{reconstruct}} = \text{argmin}_\theta \sum_i \log P(x_i|\hat{x}, \theta)$$

where θ represents the model parameters.

2. Encoder-Decoder Architecture:
BART's architecture is divided into two parts: an encoder and a decoder. The encoder processes the input text bidirectionally, capturing the full context of each token by considering its left and right contexts. This bidirectional encoding enables the model to gain a deep understanding of the input sequence. The decoder, on the other hand, is autoregressive, generating output tokens sequentially. During decoding, BART maximizes the likelihood of the next token based on previously generated tokens and the encoder's output. This is mathematically represented as:

$$P(y) = \prod_{t=1}^{T} P(y_t|y_{<t}, \text{Enc}(x))$$

where y_t is the token at position t, $y_{<t}$ are the previous tokens, and $\text{Enc}(x)$ is the encoder output.

3. Fine-Tuning for Summarization:
After pretraining, BART is fine-tuned on summarization datasets, such as CNN/DailyMail, where the model learns to generate summaries based on long-form input texts. The model parameters are adjusted using supervised learning, where the objective is to maximize the likelihood of generating target summaries Y for given inputs X. The fine-tuning loss function

is:

$$L_{\text{summ}} = \text{argmax}_\theta \sum_{(X,Y)} \log P(Y|X,\theta)$$

where X and Y are the input and summary pairs, respectively.

4. Beam Search Decoding:
During inference, BART uses a decoding strategy called beam search, which allows it to evaluate multiple potential sequences of tokens simultaneously to find the most probable output. Beam search maintains a set of top k sequences at each decoding step, where k is the beam width, helping BART produce more coherent summaries by exploring multiple candidate sequences:

$$y^* = \text{argmax}_y \sum_{t=1}^{T} \log P(y_t|y_{<t}, \text{Enc}(x))$$

Here, y^* is the sequence with the highest likelihood among the top k candidates.

5. Evaluation Metrics:
To assess the quality of the generated summaries, BART is evaluated using metrics such as ROUGE (Recall-Oriented Understudy for Gisting Evaluation). ROUGE measures the overlap of n-grams between the generated summary and the reference summary. The ROUGE score is defined as:

$$\text{ROUGE-N} = \frac{\sum_{\text{S}\in\text{RefSumm}} \sum_{\text{gram}_n\in\text{S}} \text{Count}_{\text{match}}(\text{gram}_n)}{\sum_{\text{S}\in\text{RefSumm}} \sum_{\text{gram}_n\in\text{S}} \text{Count}(\text{gram}_n)}$$

where $\text{Count}_{\text{match}}(\text{gram}_n)$ is the number of overlapping n-grams, and $\text{Count}(\text{gram}_n)$ is the total n-grams in the reference summaries.

These mechanisms allow BART to achieve high accuracy in text summarization tasks, providing coherent and contextually rich summaries suitable for practical applications in news, business, and academic fields (Wolf et al., 2020).

3.5 DistilBERT for Contextual Understanding

DistilBERT plays a critical role in capturing the semantic relationships between words in text, making it an effective model for identifying contextually significant keywords that represent the core ideas within a document. By leveraging DistilBERT's pre-trained attention mechanisms, we extract key entities, terms, and phrases most relevant to the document's main topics. The model processes each sentence to recognize the words that align with the primary themes of the document, allowing for accurate and relevant keyword extraction.

DistilBERT's similarity measurement layer uses the sentence representation output to gauge the relevance of specific words within the text. This process begins by concatenating key vectors—[CLS] for classification, QW for the question word, and EA for the expected answer—into a single vector, represented as:

$$\text{I}_\text{v} = \text{concat}([\text{CLS}], \text{QW}, \text{EA})$$

The final output layer produces a two-dimensional vector, where one element indicates relevance and the other irrelevance. This output vector is generated using weights $\text{W}_{\text{h2}} \in \mathbb{R}^{2\times 2048}$ and biases $\text{b}_{\text{h2}} \in \mathbb{R}^2$, as shown in:

$$f(q, a) = \text{softmax}(\text{W}_{\text{h2}} \cdot \text{H}_{\text{v}} + \text{b}_{\text{h2}})$$

3.5.1 Performance in the Computer Science Domain

DistilBERT has shown strong performance in extracting keywords within technical fields, particularly in Computer Science, due to its training on a vast, diverse corpus that includes scientific and technical literature. This specialized pre-training enables DistilBERT to identify domain-specific terms, technical phrases, and specialized jargon that are prevalent in CS literature. For our project, this feature was especially beneficial, as the dataset contained many documents rich in technical vocabulary. DistilBERT's ability to recognize and prioritize these terms enhanced the accuracy and relevance of the extracted keywords, aligning well with the project's requirements for handling technical content.

3.5.2 Enhancing Keyword Extraction with Stop Words and Length Limitation

To further refine the quality of keywords extracted by DistilBERT, we applied two additional techniques: stop word filtering and keyword length limitation. Stop words, such as common conjunctions, articles, and prepositions, typically add little standalone meaning. By establishing a customized list of stop words, we filtered out these less significant words, resulting in clearer and more relevant keyword extraction tailored to our project's use case.

Additionally, we implemented a length limit on the extracted keywords, setting a predefined maximum length for terms or phrases. This restriction ensured that only concise and impactful terms were retained, preventing overly long or generic keywords. This method led to a more coherent and focused keyword output, facilitating contextual search and further analysis.

By combining DistilBERT with stop word filtering and keyword length limitations, we achieved a refined keyword extraction process that accurately captures the essential content of each document. This approach significantly enhances the system's ability to provide contextually relevant insights, allowing users to efficiently identify and retrieve key information across diverse document types.

3.6 Wikipedia API for Contextual Knowledge Retrieval

The Wikipedia API is a powerful tool that allows the system to retrieve contextual information about keywords extracted from documents. By integrating Wikipedia, we can access summaries and explanations of specific terms, providing users with quick access to background information directly within the system. This functionality is especially valuable in fields where immediate, concise information on complex topics is needed.

The Wikipedia API provides a RESTful interface, allowing us to query Wikipedia's vast database for summaries, detailed articles, and metadata on specific keywords. Here's how the API can be utilized for retrieving context on document keywords:

1. Keyword Search:
The system first sends a search request using the `search` endpoint to find articles relevant to the keyword. This search returns a list of possible articles, with the most relevant article typically at the top. A typical API call to search for a keyword, say "artificial intelligence," looks like this:

```
https://en.wikipedia.org/w/api.php?action=query&list=search&srsearch=
artificial intelligence&format=json
```

This query tells the API to perform a search (`list=search`) for "artificial intelligence" and return results in JSON format.

2. Retrieving Article Summaries:
Once the top article is identified, a second request retrieves the introductory summary of the article. This extract is often sufficient for understanding the core concept of the keyword without needing the entire article. To retrieve an extract for a specific article title, such as "Artificial Intelligence," the following API call is used:

```
https://en.wikipedia.org/w/api.php?action=query&prop=extracts&exintro=
True&titles=Artificial Intelligence&format=json
```

Here, `prop=extracts` specifies that the text extract is required, and `exintro=True` limits the result to the introductory section, returning a brief, relevant summary in JSON format.

3. Parsing the Response:
The JSON response from the API includes metadata and the article's summary. The system parses the response to extract and display the relevant text alongside the keyword. A simplified response might look like this:

```
{
   "query": {
      "pages": {
         "12345": {
            "title": "Artificial Intelligence",
            "extract": "Artificial intelligence is
            the simulation of human intelligence processes
            by machines, especially computer systems..."
         }
      }
   }
}
```

Here, the `extract` field provides a brief description, which the system can present as context for the keyword.

4. Error Handling and Fallbacks:
In cases where the Wikipedia API does not return a result, the system can implement fallback methods, like expanding the search or notifying the user if no relevant article is found. Managing rate limits is also important to ensure API requests remain within allowable thresholds.

By combining keyword extraction with real-time contextual information from Wikipedia, this API integration gives users quick, relevant background information, enhancing the understanding of content and streamlining the retrieval process.

4 Solution Implementation

This section describes how to implement the solution on the front end.

4.1 Main Page

The Main Page of the system presents a streamlined layout designed to enhance the document processing experience. A navigation panel on the left allows users to move smoothly between different functions, including file uploads, initiating back-end processes, viewing detection and extraction results, generating summaries and keywords, and exploring Wikipedia links for further contextual understanding. The central display area adapts to the selected task, providing interactive elements that let users engage with each step of the process. For example, during YOLO detection, visual representations of detected text areas are displayed, while summary and keyword pages show the processed textual information. Control buttons guide users seamlessly through each phase, from file uploads to downloading results, creating an intuitive environment that supports a smooth progression through all stages of document analysis, from input to output.

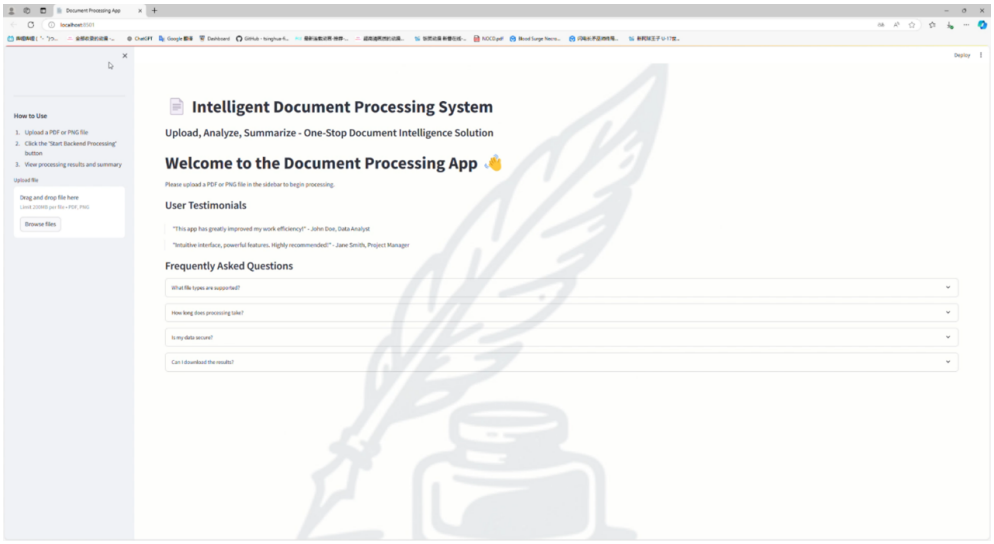

Figure 1: Main Page

4.2 Backend Starting Page

The Backend Starting Page provides a foundational interface that initiates the document processing workflow. This page allows users to select and upload documents. It provides users with clear prompts to define the file type, ensuring that the system correctly interprets the input format—whether PDF, scanned images, or other formats.

Once the document is uploaded, the backend processes automatically begin, triggering the system's advanced layout analysis, text recognition, and subsequent processing steps. The page is designed to streamline user interaction, guiding them through the essential configurations before the automated analysis pipeline takes over, ensuring a seamless and efficient document processing initiation.

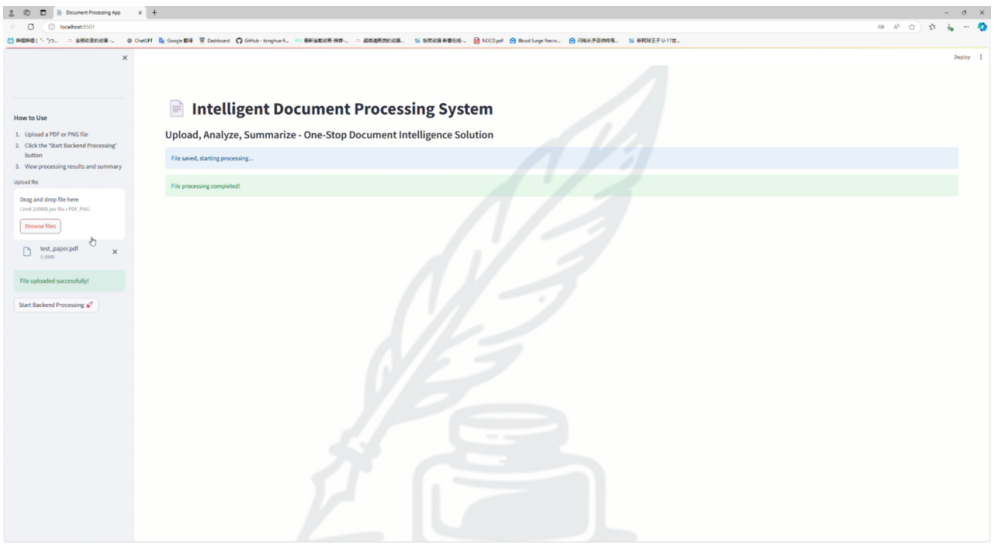

Figure 2: Backend starting page

4.3 Process Display Page

In YOLO Detection Stage, the backend begins by using the YOLO model to detect and mark document elements such as texts, headers, tables, and paragraphs. This step ensures that each part of the document is segmented accurately for further processing.

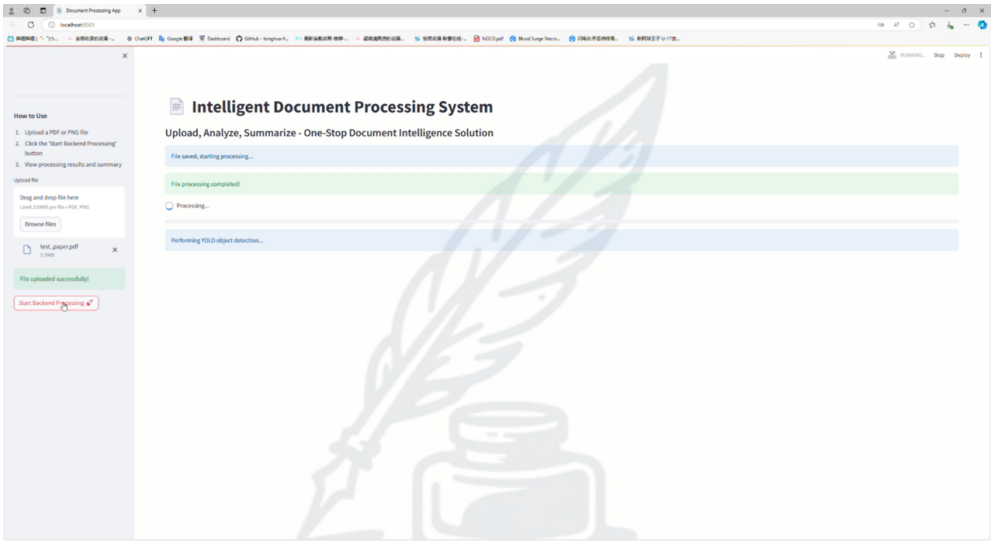

Figure 3: YOLO Detection Page

In Text Detection Stage, the progress advances to converting detected layout segments into editable text using OCR (Optical Character Recognition). This conversion enables the detected content to be used in downstream tasks like summarization.

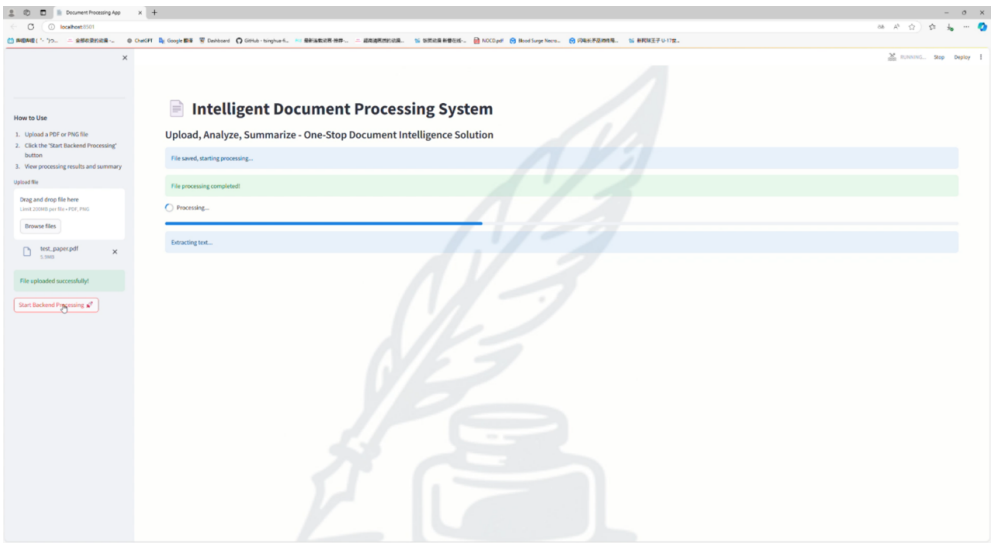

Figure 4: Text Detection Page

In Summary Generation Stage, the system processes the extracted text to generate a concise summary, capturing the document's main points. This stage is crucial for condensing lengthy information into essential insights.

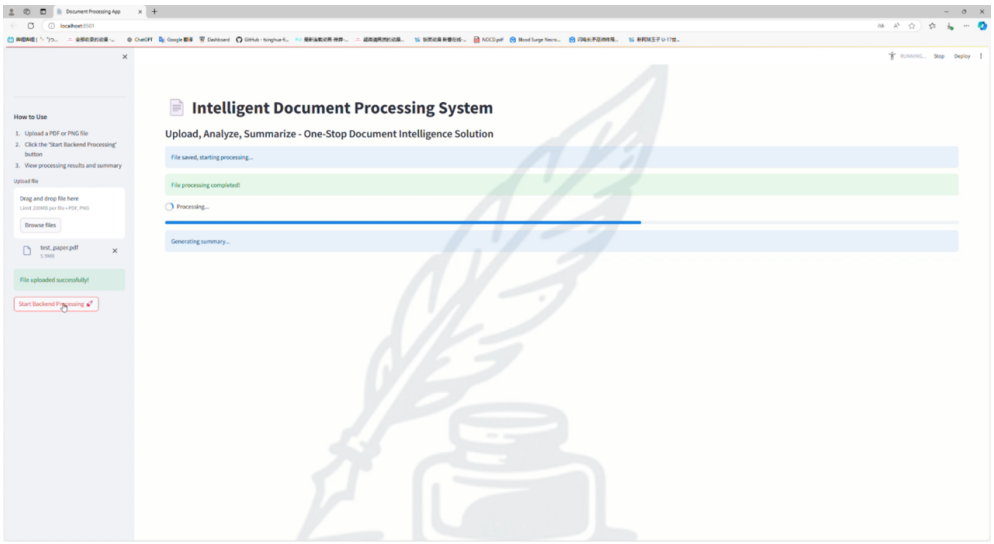

Figure 5: Summary Generation Page

In Keyword Generation Stage, key terms are identified and linked to relevant background information using Wikipedia. This final step provides users with contextual knowledge, enhancing document comprehension and enabling quick access to related topics.

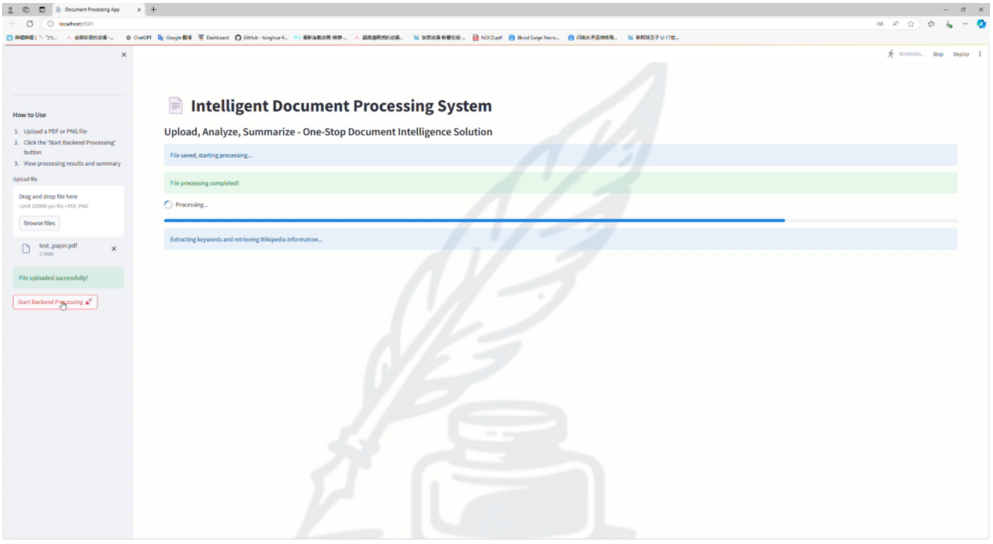

Figure 6: Keyword Generation Page

4.4 YOLO Results Browse and Download Page

In Results Browse Page, users can view a detailed display of the results generated through the document processing stages. The page is divided into distinct areas, where each result type is accessible for review.

YOLO Results Browse allows users to see the specific document layout detected by the YOLO model. Each segment, including headers, tables, and paragraphs, is highlighted, giving users an overview of how the system interpreted the document's structure.

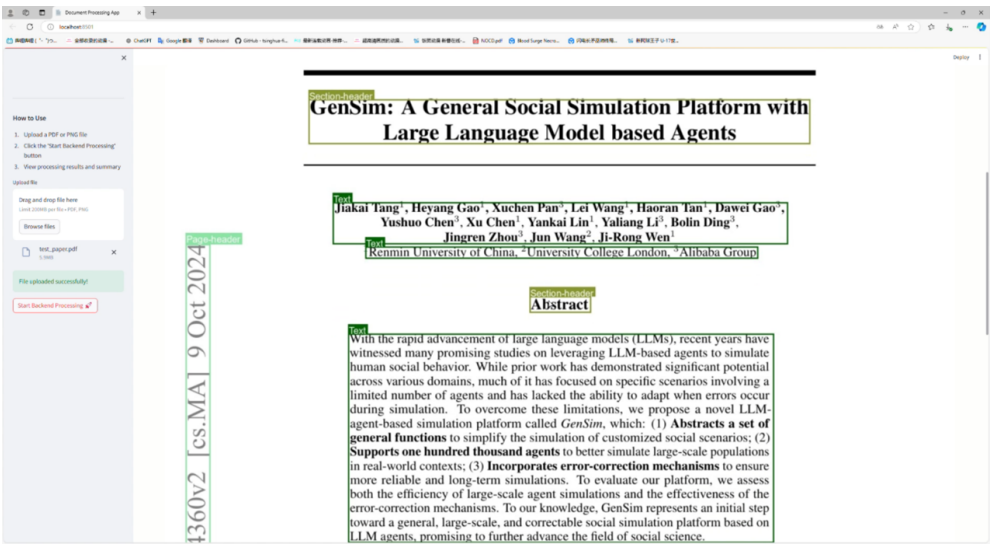

Figure 7: YOLO Results Browse Page

The Results Download Page is dedicated to enabling users to download only the YOLO detection results. This page provides a straightforward option for saving the detected layout structure, including the segmented headers, paragraphs, tables, and other document elements marked during the YOLO processing stage. By focusing specifically on YOLO results, this download page ensures users can quickly and efficiently access the structural layout data for further analysis or integration into other applications.

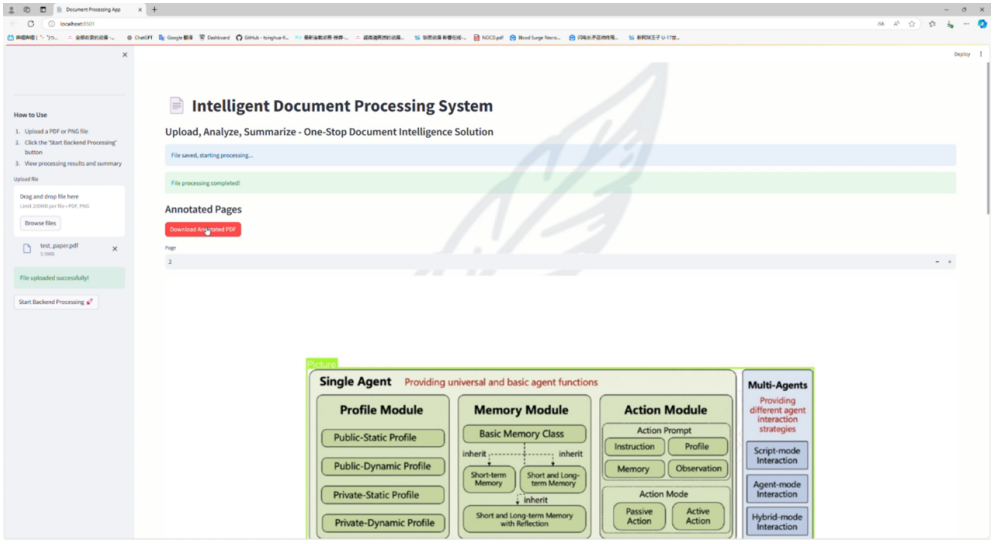

Figure 8: YOLO Results Download Page

4.5 Final Results Page

Summary and Keyword Results showcases the system-generated summaries alongside extracted keywords. This section provides a condensed view of the document content, helping users quickly grasp the main points and key terms.

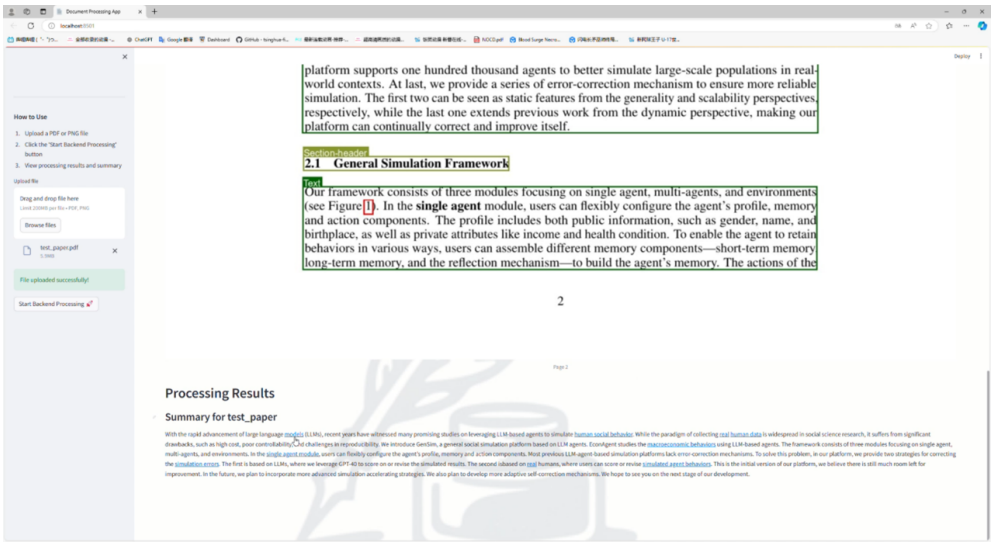

Figure 9: Summary and Keyword Results Page

Wikipedia Results links each extracted keyword to relevant background information. By connecting keywords to external knowledge sources, this part of the Results Browse Page enriches the user's understanding of document content through easy access to additional resources.

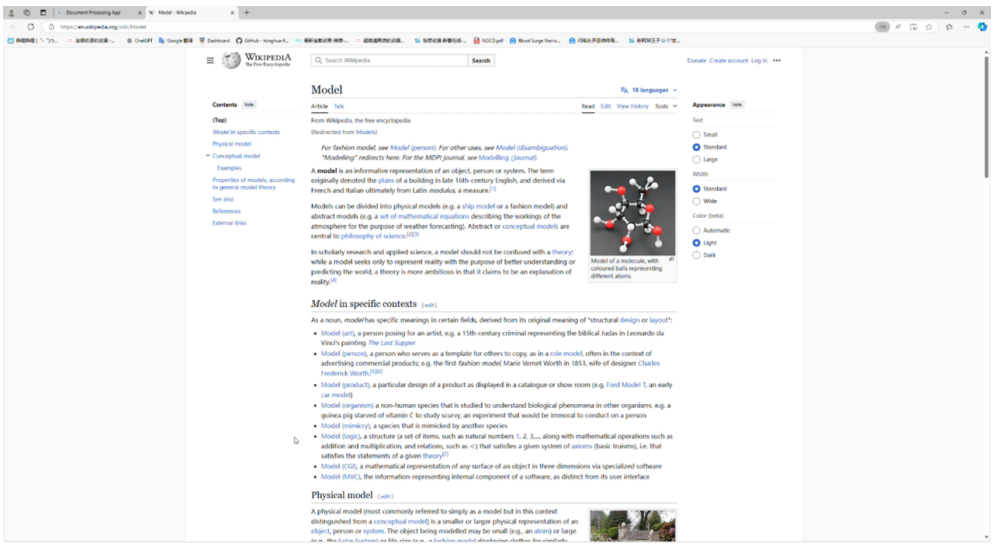

Figure 10: Wikipedia Results Page

4.6 Overall Design

The Overall Design of the document processing system integrates both frontend and backend components to create a cohesive and user-friendly experience. The frontend is designed with intuitive navigation, allowing users to move easily through different functionalities, including uploading documents, initiating backend processes, viewing YOLO-detected layouts, extracted text, summaries, and keywords. Each step provides interactive features and clear instructions, ensuring that users can complete their tasks with minimal effort.

In the backend, powerful machine learning models and tools—such as YOLOv10 for layout detection, Tesseract OCR for text extraction, BART for summarization, and DistilBERT for keyword extraction—work together to automate complex document processing tasks. Each stage of the backend pipeline feeds into the next, with initial YOLO detection creating structured segments for OCR, which then provides textual content for summarization and keyword analysis. Wikipedia API integration further enhances the user experience by linking keywords to contextual knowledge, ensuring that users can easily access relevant information.

This combination of robust backend capabilities and a user-centered frontend interface provides a highly efficient, accurate, and accessible document processing system. The design ensures that the system can handle various document formats and content complexities, making it adaptable for different user needs and applications.

5 Conclusion

This section provides a comprehensive overview of the outcomes, limitations, and future directions of the project.

5.1 Summary

The project, titled “Text Auto Detection with Background Knowledge Extraction,” aimed to develop an automated system to streamline document layout analysis, text recognition, and knowledge extraction. By leveraging cutting-edge tools such as YOLOv10 for layout detection,

Tesseract OCR for text extraction, and DistilBERT for knowledge retrieval, the system successfully transforms complex document structures into structured, searchable content. Each tool plays a critical role in converting unstructured documents into a format suitable for efficient information retrieval.

A key feature of the system is its integration with Wikipedia's API, which provides contextual information for keywords and concepts extracted from documents. This addition allows the system to enrich document content by linking relevant background knowledge, helping users understand complex terms and topics directly within the processing framework. The seamless combination of these components ensures high accuracy and efficiency, making this solution valuable for real-world applications, including academic research, legal documentation, and business intelligence.

5.2 Limitations

While the project achieved significant progress, certain limitations were identified that impact the system's overall effectiveness. The accuracy of text recognition and knowledge extraction remains highly dependent on the quality and structure of input documents. Issues such as poor image quality, non-standardized layouts, and inconsistent terminology can affect the performance of layout detection and OCR processes, potentially leading to incomplete or inaccurate text extraction. For example, documents with extensive visual clutter or low-resolution images may result in the OCR misinterpreting or missing text elements.

Another limitation is related to the distilled architecture of DistilBERT, which, while efficient, may struggle to capture highly specialized or nuanced information specific to certain fields. This constraint limits the system's ability to provide comprehensive background knowledge in domain-specific contexts, where advanced and detailed knowledge is essential. Additionally, the reliance on external data sources, such as Wikipedia's API, introduces potential challenges regarding the completeness, reliability, and currency of contextual information. If Wikipedia lacks detailed or up-to-date information on a given topic, the system's ability to offer relevant knowledge may be compromised.

5.3 Future Plan

Future work will focus on addressing the identified limitations to enhance system performance and broaden its application scope. One priority will be expanding the dataset to include a wider range of document formats and content specific to various professional fields, improving the system's ability to handle diverse layouts and terminologies.

In addition, the project aims to explore integration with other external knowledge sources beyond Wikipedia, potentially incorporating academic databases or industry-specific resources. This expansion will improve the system's capacity to provide accurate and comprehensive background information for highly specialized fields. Furthermore, optimizing the architecture to enhance scalability and reduce resource consumption will be prioritized, allowing the system to operate efficiently across different devices and user environments. These future developments will make the system even more versatile, reliable, and applicable in a variety of professional and academic contexts.

References

Lewis, M. (2019). Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *arXiv preprint arXiv:1910.13461*.

Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You only look once: Unified, real-time object detection. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Smith, R. (2007). An overview of the tesseract ocr engine. *Ninth International Conference on Document Analysis and Recognition (ICDAR 2007)*, *2*, 629–633. https://doi.org/10.1109/ICDAR.2007.4376991

Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M., Davison, J., Shleifer, S., von Platen, P., Ma, C., Jernite, Y., Plu, J., Xu, C., Le Scao, T., Gugger, S., ... Rush, A. (2020, October). Transformers: State-of-the-art natural language processing. In Q. Liu & D. Schlangen (Eds.), *Proceedings of the 2020 conference on empirical methods in natural language processing: System demonstrations* (pp. 38–45). Association for Computational Linguistics. https://doi.org/10.18653/v1/2020.emnlp-demos.6