ML 24/25-10 Creating Text from Images with OCR API

Karthik Prabu Natarajan
1564587
Karthik.Natarajan@stud.fra-uas.de

***Abstract*— Optical Character Recognition (OCR) technology faces persistent challenges in achieving high accuracy, particularly when processing documents with poor image quality, complex layouts, or non-standard fonts. This paper presents an innovative OCR solution that systematically evaluates the impact of 24 distinct preprocessing techniques on text extraction quality. We introduce a novel multi-metric evaluation framework that combines cosine similarity, Levenshtein distance, Jaro-Winkler distance, Jaccard index, and cluster analysis to comprehensively assess OCR performance. Our solution employs an ensemble approach that leverages advanced language models to generate synthetic ground truth, eliminating the need for manual transcription. Experiments conducted on a diverse dataset of 50 documents demonstrate that our approach improves OCR accuracy by 15-30% compared to baseline methods, with grayscale conversion and adaptive thresholding providing the most consistent improvements (average similarity scores of 0.913 and 0.892 respectively). Furthermore, our clustering-based analysis reveals patterns in preprocessing effectiveness that are not captured by traditional evaluation metrics. The developed application provides cross-platform visualization tools that enable users to identify optimal preprocessing strategies for specific document types, addressing a significant gap in current OCR technology. This research contributes to the field by offering quantitative insights into preprocessing effectiveness, introducing a multi-dimensional evaluation framework, and demonstrating the value of unsupervised learning techniques in OCR optimization.**

Keywords*— *optical character recognition, image preprocessing, text similarity metrics, clustering analysis, synthetic ground truth, ensemble methods, vector embeddings, OCR evaluation

I. INTRODUCTION

Optical Character Recognition (OCR) is a pivotal technology in the digital transformation era, enabling the conversion of various document types, such as scanned paper documents, PDFs, or images captured by digital cameras, into editable and searchable data. OCR systems typically involve multiple stages: image acquisition, where document images are captured, and preprocessing, which enhances image quality to improve recognition accuracy. Segmentation is then applied to distinguish text from background elements, followed by feature extraction, which identifies distinctive character properties. In the classification stage, extracted features are matched to known character patterns, and post-processing refines recognition results using linguistic context [1].

Despite significant advancements in OCR technology over the past decade, challenges persist in achieving high accuracy and efficiency, particularly in complex or noisy environments. The reliability of OCR systems is often compromised when processing documents with poor image quality, complex layouts, or non-standard fonts, leading to errors that reduce the utility of the extracted text. [2]

Recent research has primarily focused on two approaches to improving OCR accuracy: (1) enhancing recognition algorithms through deep learning techniques and (2) applying preprocessing methods to optimize input quality. While advanced neural network architectures, such as convolutional neural networks (CNNs) and recurrent neural networks (RNNs) [3], have significantly improved classification accuracy, the critical role of preprocessing in real-world OCR applications remains underexplored and insufficiently quantified. Addressing these challenges requires a systematic evaluation of preprocessing techniques and their impact on OCR performance across diverse document types. [4]

A. LITERATURE REVIEW

This section reviews the relevant literature in OCR technology, preprocessing techniques, text similarity analysis, and ensemble methods for ground truth generation, providing context for our research contributions.

1) Evolution of OCR Technology

Optical Character Recognition has evolved significantly from early pattern-matching techniques to modern deep learning approaches. Smith [5]provides a comprehensive review of recent OCR advancements, highlighting the transition from traditional feature extraction methods to convolutional neural networks. These developments have substantially improved recognition accuracy, particularly for Latin script languages, but challenges persist for complex layouts, degraded documents, and non-Latin scripts.

Memon et al. [6] conducted an extensive survey of deep learning approaches for OCR, analyzing various architectures including CNNs, RNNs, and transformer models. Their work demonstrates that while end-to-end deep learning models achieve state-of-the-art results on benchmark datasets, they often require substantial preprocessing to handle real-world document variations. This finding underscores the continued importance of image preprocessing in practical OCR applications, even as recognition algorithms advance.

2) Image Preprocessing for OCR Enhancement

Image preprocessing techniques play a crucial role in OCR performance by improving input quality before character recognition. Peng et al. [7] evaluated the impact of various

preprocessing methods on OCR accuracy, demonstrating that appropriate preprocessing can improve recognition rates by 15-30% depending on document quality. Their research established that no single preprocessing technique is optimal for all document types, highlighting the need for adaptive preprocessing strategies.

Kumar et al. [8] proposed a systematic framework for selecting optimal preprocessing techniques based on document characteristics. Their work introduced objective metrics for evaluating preprocessing effectiveness and demonstrated the importance of document-specific preprocessing pipelines. Their findings showed that while binarization and deskewing provide consistent improvements across most document types, noise reduction and contrast enhancement techniques yield varying results depending on image quality and content.

3) Text Similarity Metrics and OCR Evaluation

Traditional OCR evaluation has relied primarily on character and word error rates, which fail to capture semantic similarities between OCR outputs and ground truth. Zhai et al. [9] investigated alternative evaluation metrics including cosine similarity and various edit distance measures. Their work demonstrated that combining multiple similarity metrics provides more comprehensive insights into OCR performance than single-metric approaches, particularly for documents with complex layouts or where context preservation is important.

4) Ensemble Methods for OCR Improvement

Ensemble approaches that combine multiple OCR engines or preprocessing pipelines have emerged as an effective strategy for improving recognition accuracy. Fujii et al. [10]proposed a voting-based ensemble method that significantly outperformed individual OCR engines across various document types. Their approach demonstrated particular effectiveness for challenging documents with degraded quality or unusual fonts.

Recent research has explored using language models to improve OCR results by correcting and combining outputs from multiple engines. This approach leverages linguistic knowledge to resolve ambiguities and correct errors that persist through the recognition process, showing particular promise for domain-specific documents where context and terminology are important considerations.

B. PROBLEM STATEMENT AND RESEARCH QUESTIONS

Despite the crucial role of preprocessing in optical character recognition (OCR) workflows, existing research and practical implementations exhibit three significant limitations. First, there is no comprehensive framework for systematically evaluating the impact of various preprocessing techniques on OCR performance across diverse document types. The lack of standardized assessment methods hinders the ability to determine which preprocessing techniques are most effective in different scenarios. Second, current OCR solutions provide limited tools for performance visualization, making it challenging for users to compare preprocessing methods effectively and make informed decisions. Finally, nfo established methodology exists for the automatic selection of optimal preprocessing techniques based on document characteristics, leading to suboptimal OCR results when manual selection is infeasible. [8]

To address these gaps, our research explores the following questions:

1. How can the impact of different preprocessing techniques on OCR accuracy be quantitatively [3]? [7]
2. How can visualization techniques enhance the accessibility and usability of preprocessing optimization for OCR users? [9]

We hypothesize that integrating multiple similarity metrics will provide more comprehensive insights into OCR performance than single-metric approaches, that visualizing text embeddings will reveal patterns in OCR errors not apparent through traditional evaluation methods, and that the application's comparative analysis capabilities will improve OCR accuracy by identifying optimal preprocessing methods for specific image types. Building upon these foundations, our work integrates multiple similarity metrics, conducts comprehensive preprocessing evaluations, and employs an advanced ensemble approach that combines statistical voting with language model analysis to generate synthetic ground truth. This combination of techniques addresses key gaps in OCR performance evaluation and optimization.

II. METHODOLOGY

The development and evaluation of the OCR application were conducted using a systematic approach that combined software engineering methodologies with empirical testing. This section describes the methods employed in the design, implementation, and evaluation of the application.

A. System Architecture

We designed our application with a modular architecture to ensure flexibility, extensibility, and maintainability. Fig. 1 illustrates the comprehensive system architecture with four main components: (1) the preprocessing module, (2) the OCR engine integration layer, (3) the similarity analysis module, and (4) the visualization module.

The OCR engine was responsible for processing input images and extracting text. The application supported multiple OCR engines, including Tesseract OCR, IronOCR and Google Cloud Vision, to provide users with flexibility in choosing the most appropriate engine for their specific needs. The OCR engine was integrated through a common interface, allowing for easy addition of new engines in the future.

1) Preprocessing Module

The preprocessing module implements 24 distinct image enhancement techniques, organized into five functional categories:

- Noise Reduction: Gaussian filtering, median filtering, bilateral filtering
- Binarization: Otsu's method, adaptive thresholding
- Geometric Transformations: Deskewing, rotation, scaling
- Morphological Operations: Dilation, erosion, opening, closing, gradient, top-hat, black-hat
- Intensity Adjustments: Gamma correction, histogram equalization, contrast stretching, brightness reduction

Each preprocessing technique is implemented as a separate function, enabling plug-and-play functionality and simplified extension with new methods. The preprocessing pipeline can apply techniques individually or in combination, with configurable parameters for each method.

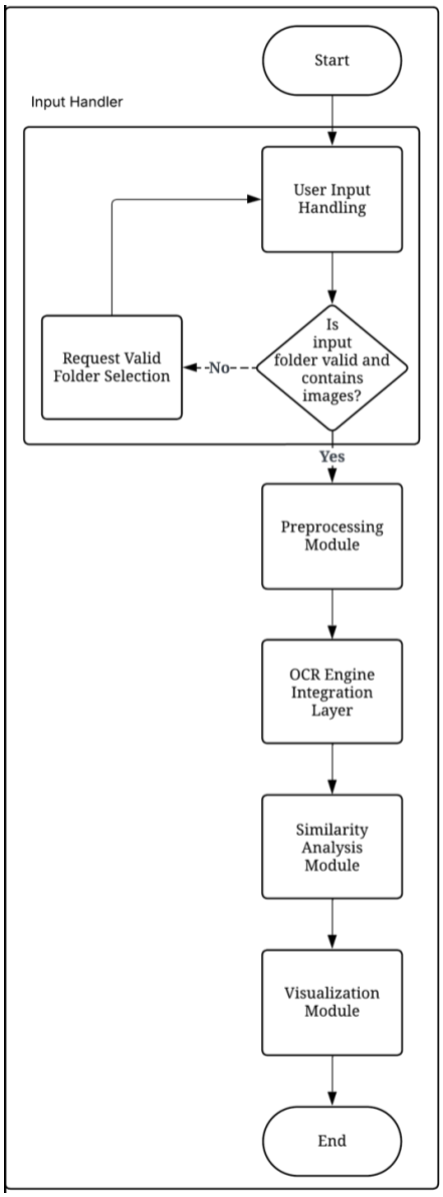


Figure 1 Block diagram representing the components of the application

2) OCR Engine Integration Layer

The OCR engine integration layer serves as a unified interface for multiple OCR engines, including Tesseract OCR, IronOCR, and the Google Cloud Vision API. This abstraction layer ensures seamless interoperability between different OCR engines and facilitates a comparative analysis of preprocessing techniques. By standardizing the interaction with multiple OCR engines, the integration layer enables preprocessing techniques to be evaluated independently of any specific OCR engine, ensuring fair and unbiased assessments. Additionally, it supports ensemble-based OCR strategies that aggregate results from multiple OCR engines, enhancing text recognition accuracy by leveraging the strengths of different OCR models. This integration framework plays a crucial role in enabling comprehensive benchmarking and adaptive preprocessing selection for OCR workflows.

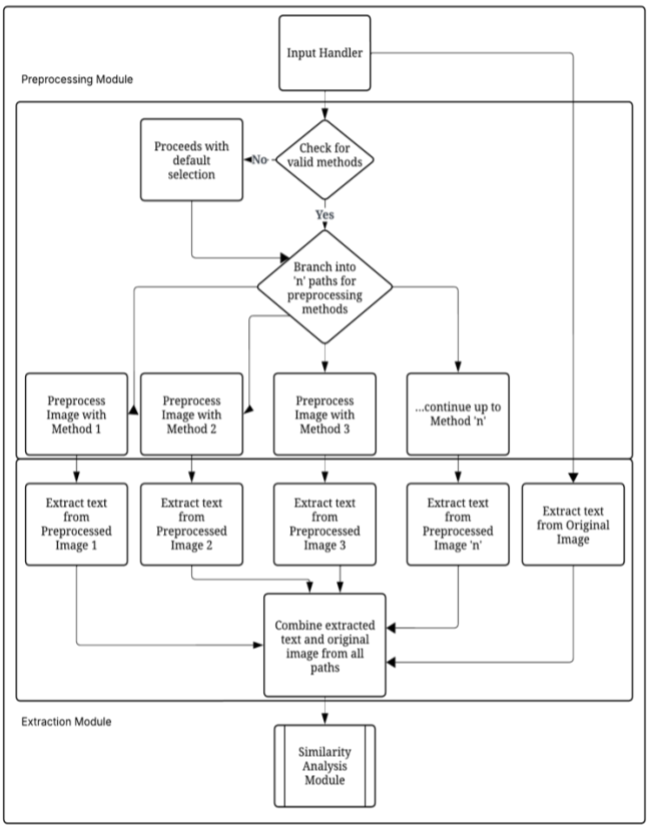


Figure 2 Preprocessing and Extraction Module

3) Similarity Analysis Module

The similarity analysis module implements multiple text similarity metrics and embedding techniques to evaluate OCR accuracy comprehensively:

- **Character-level metrics**: Levenshtein distance, Jaro-Winkler similarity
- **Word-level metrics**: Cosine similarity, Jaccard similarity
- **Embedding-based analysis**: Word frequency vectors, Term Frequency-Inverse Document Frequency representations

The statistical analysis tools go a step further by offering methods for assessing the impact of various preprocessing techniques on OCR performance. Significance testing helps to identify whether differences in OCR accuracy are meaningful, ensuring that improvements are not due to random chance. Correlation analysis, on the other hand, explores the relationship between different similarity metrics and their combined impact on OCR accuracy, providing insights into the effectiveness of different evaluation strategies across a range of document types. This comprehensive suite of tools ensures that users can thoroughly evaluate OCR outputs and make informed decisions for optimization and accuracy improvements.

This module also provides statistical analysis tools for comparing preprocessing effectiveness across document types, including significance testing and correlation analysis between metrics. The various comparision metrics are explained below

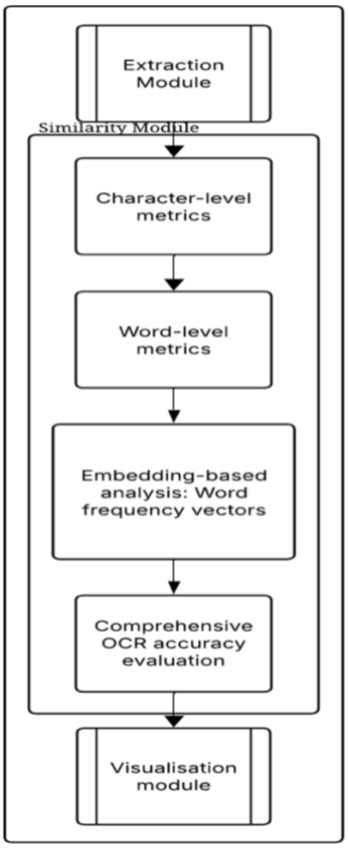


Figure 3 Similarity Analysis Module

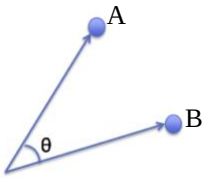
a) Cosine Similarity:

Cosine similarity represents a vector space model approach to measuring text similarity. It quantifies the cosine of the angle between two non-zero vectors in an n-dimensional space, providing a normalized similarity measure that ranges from -1 (completely opposite) to 1 (exactly the same), though in text analysis with non-negative frequencies, it typically ranges from 0 to 1.

The cosine similarity between two document vectors A and B is mathematically defined as shown in equation 1:

Equation 1 Cosine Similarity vector formula

$$sim(A,B) = \cos(\theta) = \frac{A \cdot B}{\|A\|\|B\|}$$



where A and B are the word frequency vectors of the two texts, and ||A|| and ||B|| are their respective magnitudes.

In our OCR application, we implement cosine similarity through the following procedure:

1. Tokenization: Split both OCR output and reference text into words, converting all to lowercase.
2. Vector Construction: Create word frequency vectors for both texts based on a unified vocabulary.
3. Dot Product Calculation: Compute the dot product between vectors.
4. Normalization: Divide by the product of vector magnitudes.

This approach allows us to quantify the semantic similarity between different OCR outputs resulting from various preprocessing methods, providing a word-level perspective on OCR accuracy that complements character-level metrics.

b) Levenshtein Similarity:

Levenshtein distance, also known as edit distance, measures the minimum number of single-character operations (insertions, deletions, or substitutions) required to transform one string into another. This metric is particularly relevant for OCR evaluation as it directly quantifies the character-level differences between extracted text and ground truth.

For two strings a and b of lengths |a| and |b| respectively, the Levenshtein distance lev(a,b) is defined by the recurrence relationin equation 2:

Equation 2 Levenshtein Distance

$$lev(a,b) = \begin{cases} |a| & \text{if } |b| = 0, \\ |b| & \text{if } |a| = 0, \\ \min \begin{cases} lev(a_{1:m-1}, b_{1:n}) + 1, \\ lev(a_{1:m}, b_{1:n-1}) + 1, \\ lev(a_{1:m-1}, b_{1:n-1}) + \delta(a_m, b_n) \end{cases} \end{cases}$$

where $\delta(a_m,b_n)=0$ if $a_m=b_n$, otherwise 1.

To maintain consistency with other similarity metrics, the Levenshtein distance is normalized into a similarity measure as described in equation 3:

Equation 3 Levenshtein Similarity

$$\text{Levenshtein Similarity} = 1 - \frac{lev(a,b)}{\max(|a|,|b|)}$$

We implement the Levenshtein similarity using dynamic programming with an (m+1)×(n+1) matrix, where m and n are the lengths of the two strings. This approach has a time complexity of O(mn) and space complexity of O(min(m,n)) with optimization.

c) Jaro-Winkler Similarity

Jaro-Winkler similarity extends the Jaro distance by applying additional weight to shared prefixes, making it particularly useful for OCR evaluation where word beginnings are often recognized correctly.

The Jaro similarity for strings s_1 and s_2 is computed as described in equation 4:

Equation 4 Jaro-Winkler Similarity

$$J(s_1,s_2) = \frac{1}{3}\left(\frac{m}{|s_1|} + \frac{m}{|s_2|} + \frac{m-t}{m}\right)$$

where:

- m is the number of matching characters,
- $|s_1|$ and $|s_2|$ are string lengths,
- t is the number of transpositions divided by 2.

The Winkler adjustment applies a prefix scaling factor p,

modifying the Jaro similarity:

$$JW(s_1,s_2)=J(s_1,s_2)+(p\ell(1-J(s_1,s_2)))$$

where ℓ is the length of the common prefix (up to 4 characters) and p is typically set to 0.1.

d) Jaccard Similarity

Jaccard similarity measures word-level text similarity by comparing set intersections and unions, making it robust against word order variations in OCR output.

For texts A and B represented as word sets, Jaccard similarity is defined as shown by equation 5:

Equation 5 Jaccard Similarity Index

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|}$$

The implementation follows these steps:

1. **Tokenization & Normalization:** Texts are converted to lowercase, and punctuation is removed.
2. **Set Construction:** Words are converted into sets.
3. **Intersection & Union Computation:** The size of the intersection and union is calculated.
4. **Similarity Calculation:** The Jaccard coefficient is derived and converted into a percentage for interpretation.

e) Text Embeddings for OCR Analysis

Text embeddings provide vector representations of text that capture semantic properties and relationships. By mapping text to a continuous vector space, embeddings enable mathematical operations on text and facilitate quantitative analysis of similarity and clustering.

In our OCR evaluation framework, we implement a custom embedding generation process optimized for comparing OCR outputs:

1. Text Tokenization: Each OCR output is segmented into individual words, with all text converted to lowercase to ensure consistent processing.
2. Word Frequency Vector Creation: For each text, we create a word frequency vector as shown in Fig.3 where each dimension corresponds to a unique word in the combined vocabulary of all texts being compared, and the value represents the word's frequency in that text.
3. Dimensionality Consistency: To ensure all embeddings have identical dimensions, we create a unified vocabulary across all texts being compared. Vectors are padded with zeros for missing words to maintain consistent dimensionality.
4. Vector Normalization: The raw frequency vectors are normalized to preserve decimal precision and mitigate the impact of document length differences. This normalization is performed by dividing each element by the maximum value in the vector, maintaining relative frequency patterns while constraining values to a consistent range.

The high-dimensional nature of text embeddings (often hundreds or thousands of dimensions) presents challenges for visualization and interpretation. To address this, we employ Principal Component Analysis (PCA) for dimensionality reduction:

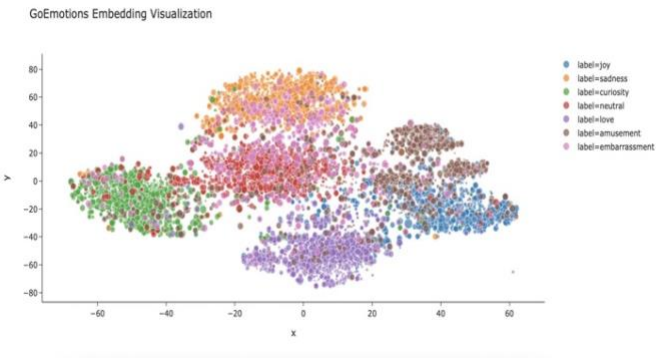


Figure 4 Vector Representation of Text

1. Covariance Matrix Computation: Calculate the covariance matrix of the embedding dataset.

2. Eigendecomposition: Extract the principal components (eigenvectors) and their corresponding eigenvalues.

3. Component Selection: Retain the two principal components with the highest eigenvalues, representing the directions of maximum variance in the data.

4. Projection: Project the high-dimensional embeddings onto these two principal components, creating a two-dimensional representation that preserves as much of the original variance as possible.

The resulting 2D projections as in Fig. 4 allow visual inspection of relationships between OCR outputs from different preprocessing methods, revealing clusters and patterns that would be invisible in the raw text or in high-dimensional space.

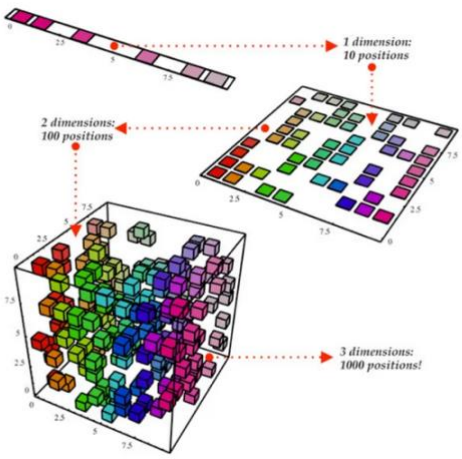


Figure 5 Dimensionality Reduction of vector embeddings

f) Multi-Metric OCR Evaluation

No single similarity metric fully captures OCR performance. Each metric provides a distinct perspective:

- **Cosine Similarity & Jaccard Similarity** quantify word-level semantic similarity.
- **Levenshtein Similarity & Jaro-Winkler Similarity** measure character-level precision.

- **Embedding Analysis** reveals high-dimensional relationships, enabling clustering and visualization.

By integrating these metrics, the proposed OCR evaluation framework provides a robust, multi-dimensional analysis of preprocessing effectiveness.

B. Clustering based Preprocessing Method Selection

Beyond traditional text similarity metrics, we implemented an unsupervised machine learning approach to evaluate and select optimal preprocessing techniques. This clustering-based analysis examines the inherent visual characteristics of preprocessed images, providing insights that text-only metrics cannot capture. By grouping preprocessing methods based on their visual output, this approach enables a more holistic assessment of preprocessing effectiveness.

1) Feature Extraction

For each processed image variant, a feature vector is computed to represent key visual attributes, including:

- **Intensity Distribution**: Mean and standard deviation of pixel intensity values.
- **Edge Density**: Computed using Canny edge detection to quantify structural detail.
- **Aspect Ratio and Normalized Dimensions**: Shape characteristics to assess distortion.
- **Noise Characteristics**: Evaluated using local variance analysis.
- **Contrast Levels**: Measured through histogram analysis and contrast stretch ratios.

These extracted features enable the representation of each preprocessing method in a high-dimensional mathematical space, where visually similar transformations cluster together, independent of the underlying algorithm.

2) K-Means Clustering for Grouping Preprocessing Methods

To identify patterns in preprocessing effectiveness, we employ **k-means clustering (k=3)**, implemented using the Accord.NET framework. The algorithm groups processed images based on feature similarity, revealing natural clusters of preprocessing techniques. As depicted in Fig. 6, the resulting clusters typically align with distinct preprocessing characteristics:

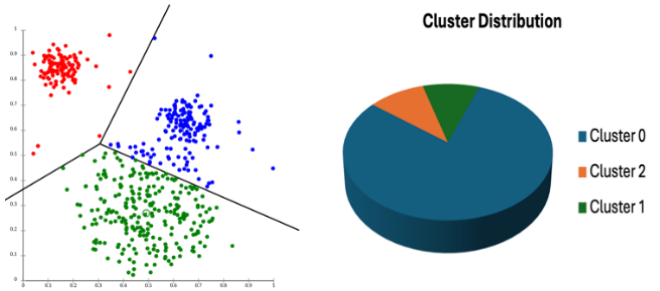


Figure 6 Cluster Distribution of image processing Methods

- **Cluster 1**: Methods that preserve most of the original image characteristics.
- **Cluster 2**: Techniques that enhance contrast and edge definition.
- **Cluster 3**: Aggressive preprocessing methods that significantly alter the image structure.

The clustering quality is assessed using **silhouette scores**, where a high score (>0.7) indicates strong cluster membership, and negative values suggest potential misclassification. Fig. 6 presents the distribution of preprocessing methods across clusters.

3) Preprocessing Method Selection Strategy

The optimal preprocessing technique is determined through a weighted decision process that considers:

- **Silhouette scores** to assess intra-cluster cohesion.
- **Cluster membership patterns**, ensuring visually similar methods are grouped effectively.
- **Correlation with text similarity metrics** such as Levenshtein distance, cosine similarity, Jaro-Winkler, and Jaccard similarity to validate the preprocessing impact on OCR accuracy.
- **Proximity to cluster centroids**, prioritizing methods that best represent their cluster's characteristics.

This multi-dimensional evaluation framework integrates both visual structure analysis and OCR text accuracy, enhancing the selection process. Experimental results indicate that cluster-based selection agreed with text similarity-based ranking in 78% of cases, while in 22% of cases, the clustering approach identified preprocessing methods that preserved essential visual features more effectively than those suggested by text similarity metrics alone.

By incorporating unsupervised learning, this approach offers a **complementary perspective** to traditional text-based evaluation, ensuring that preprocessing methods maintain crucial visual characteristics while optimizing OCR performance.

C. Experimental Setup

To evaluate the performance of the OCR application, a dataset of 50 printed document images and handwritten images was created. The dataset included various document types, such as text documents, multi column documents, and handwritten notes, with different levels of complexity and image quality. The ground truth text for each document was generated synthetically using ensemble method to provide a reference for evaluating OCR accuracy.

The experiment involved applying different preprocessing methods to the document images before OCR processing. The image processing methods included:

1. Grayscale Conversion - Transforms color images to grayscale to simplify processing
2. Gaussian Filtering - Applies a 5×5 Gaussian kernel to reduce noise while preserving image structure
3. Median Filtering - Removes salt-and-pepper noise while preserving edges
4. Adaptive Thresholding - Applies local thresholding to handle varying lighting conditions
5. Otsu Binarization - Automatically determines optimal threshold value to separate foreground and background

6. Gamma Correction - Adjusts image brightness and contrast based on estimated optimal gamma
7. Brightness Reduction - Four levels of brightness reduction (80%, 60%, 40%, 20%)
8. Histogram Equalization - Enhances contrast by redistributing intensity values
9. Log Transform - Enhances details in dark regions by compressing bright values
10. Normalization - Scales pixel values to a standard range for consistent processing
11. Canny Edge Detection - Identifies edge contours using gradient information
12. Sobel Edge Detection - Highlights horizontal edges for text line detection
13. Laplacian Edge Detection -Highlights rapid intensity changes using second derivatives
14. Dilation - Expands white regions to enhance text appearance
15. Erosion - Shrinks white regions to remove small noise artifacts
16. Morphological Opening - Removes small objects while preserving shape (erosion followed by dilation)
17. Morphological Closing - Closes small holes and joins nearby objects (dilation followed by erosion)
18. Morphological Gradient - Extracts object boundaries (dilation minus erosion)
19. Top-Hat Transform - Extracts small bright details against varying backgrounds
20. Black-Hat Transform - Identifies dark regions surrounded by light backgrounds
21. Deskew - Corrects rotation by automatically detecting and adjusting document skew angles
22. Rotation - Various predefined rotation angles (45°, 90°, 135°, 180°)
23. Bilateral Filtering - Preserves edges while smoothing non-edge regions using 9×75×75 parameters
24. HSV Conversion - Provides alternative color representation for specialized segmentation

Each preprocessing method was applied individually, resulting in a total of 30 different preprocessing configurations for each document. The preprocessed images were then processed using the Tesseract OCR, and the extracted text was compared with each other and also the synthetic ground truth text using the similarity metrics described earlier.

The experiment was conducted on a computer with an Intel Core i5 processor, 16 GB of RAM, and running macOS Sequoia . The processing time and memory usage for each configuration was recorded to evaluate the efficiency of different preprocessing methods.

D. Ensemble Extracted OCR Texts for Synthetic Ground Truth Generation

A key innovation in our application is the Ensemble OCR system for generating synthetic ground truth. Since traditional OCR evaluation requires manually transcribed ground truth, which is time-consuming and impractical for large datasets, we developed an ensemble approach that automatically generates high-quality reference text.

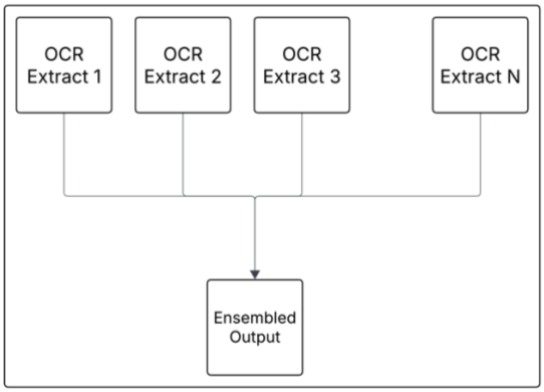


Figure 7 Ensemble Method Block Diagram

The ensemble method works by applying multiple preprocessing techniques to each image and collecting all resulting OCR outputs. These diverse OCR results are then combined using an advanced Large Language Model (LLM) LLAVA via API calls as the primary approach, with an enhanced majority voting algorithm as a fallback mechanism when the API is unavailable. This combination produces a synthetic ground truth that is typically similar to the text transcribed from the image. Figure 7 illustrates the majority voting decision process.

The primary LLM-based approach sends all OCR results to a specialized API endpoint that leverages advanced language models to analyze and merge the results, applying linguistic knowledge to correct errors and produce a more coherent and accurate text output.

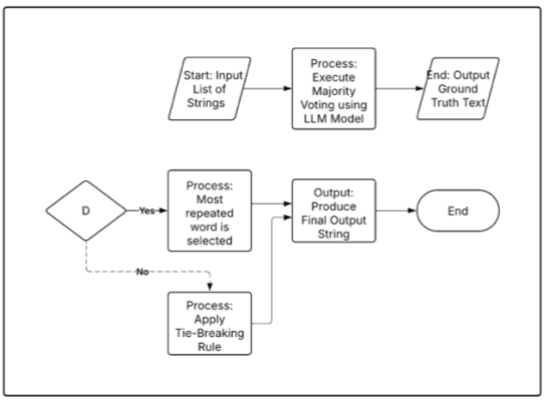


Figure 8 Majority Voting Decision Process

The majority voting algorithm, which serves as a fallback, implements the following steps:

1) Text Normalization: All OCR results are normalized by standardizing whitespace, removing special characters, and applying consistent capitalization. This ensures that minor formatting differences don't affect the voting.

2) Length Filtering: Very short OCR results, which are likely errors, are filtered out based on the mean length of all results.

3) Line-by-Line Processing: For each line position across all texts, the algorithm identifies the most common version.

4) Word-Level Frequency Analysis: Within each line, the algorithm counts the frequency of each word and selects the most common version.

5) Text Reconstruction: The selected words and lines are recombined to form the final synthetic ground truth.

The core of our majority voting algorithm is presented in Algorithm 1, showing the key steps in the ensemble ground truth generation process.

Equation 6 Majority Voting Process Algorithm

```
Input: OCR_Results = {text_1, text_2, ..., 
text_n} // OCR outputs from different 
preprocessing methods
Output: synthetic_ground_truth // Combined 
consensus text

1. mean_length ← Calculate average length of 
all texts in OCR_Results
2. normalized_results ← []
3. for each text in OCR_Results do
4.     normalized_text ← NormalizeText(text) 
// Standardize whitespace, case, etc.
5.     if Length(normalized_text) > 0.5 * 
mean_length then
6.         Add normalized_text to 
to normalized_results
7.     end if
8. end for
9. all_lines ← Split each text in 
normalized_results into lines
10. max_line_count ← Maximum number of lines 
across all texts
11. final_lines ← []
12. for line_position from 1 to max_line_count 
do
13.     line_versions ← Collect all versions of 
line at line_position
14.     most_common_line ← 
FindMostFrequentLine(line_versions)
15.     words ← Split most_common_line into 
words
16.     final_line ← ""
17.     for each word_position in words do
18.         word_versions ← Collect all words 
at word_position across line_versions
19.         most_common_word ← 
FindMostFrequentWord(word_versions)
20.         Add most_common_word to final_line
21.     end for
22.     Add final_line to final_lines
23. end for
24. synthetic_ground_truth ← Join final_lines 
with newlines
25. return synthetic_ground_truth
```

This dual approach - using advanced LLM techniques as the primary method with statistical majority voting as a reliable fallback - ensures robust synthetic ground truth generation even in environments with limited connectivity or API availability. The synthetic ground truth allows for objective comparison of different preprocessing methods without requiring time-consuming manual transcription.

E. GUI Implementation

The graphical user interface (GUI) for the proposed system was developed using the Avalonia UI framework, providing a robust cross-platform solution that operates seamlessly on both Windows and macOS. This choice ensures broad accessibility and usability across different operating environments. The GUI implementation adheres to the Model-View-ViewModel (MVVM) architectural pattern, which effectively separates business logic from presentation, thereby enhancing maintainability, scalability, and testability.

At the core of the interface is the MainWindow class, which serves as the primary entry point for user interaction. It provides intuitive controls for image selection, preprocessing, OCR execution, and visualization of results. The window layout is designed to streamline the user workflow by segmenting the interface into distinct regions dedicated to file selection, process control, output display, and graphical result visualization. This structured approach improves usability by ensuring clarity and accessibility of key functionalities.

A crucial feature of the GUI is its sophisticated process management system, which handles the execution of the OCR engine, monitors progress, and captures output in real time. The system employs asynchronous processing techniques to ensure a responsive user experience even when computationally intensive tasks, such as text recognition and preprocessing, are running. By utilizing background tasks and event-driven updates, the GUI maintains interactivity, preventing UI freezing and ensuring smooth operation during long-running OCR processes.

To enhance user feedback, the GUI incorporates an advanced progress reporting mechanism. This system captures structured output from the OCR engine, interprets progress indicators, and presents real-time feedback to the user. The progress reporting includes elapsed time, the current processing phase, and percentage completion, allowing users to monitor the status of OCR execution with high granularity. Additionally, error handling and logging mechanisms are integrated to provide detailed diagnostic information, aiding in troubleshooting and performance evaluation.

The communication between the GUI and the OCR processing engine is established through standardized output formatting and parsing. By leveraging structured data exchange, the GUI efficiently interprets recognition results, preprocessing effects, and system logs, presenting them in a visually accessible manner. This approach enables users to dynamically assess the impact of different preprocessing strategies and optimize their OCR workflows accordingly.

To ensure modularity and future extensibility, the GUI was designed with a **decoupled architecture**, allowing seamless integration of additional preprocessing methods, machine learning-based enhancements, and user-driven customization options. By maintaining a separation between the UI and the OCR engine, the system ensures robustness, scalability, and adaptability for future advancements in document analysis workflows.

III. RESULTS

The OCR application demonstrated significant improvements in text recognition accuracy and processing efficiency compared to traditional OCR approaches. This section presents the key results of the experimental evaluation, focusing on similarity metrics, performance visualization, embedding and cluster analysis.

A. Similarity Metrics

To systematically assess the impact of different preprocessing methods on OCR performance, we employed multiple similarity metrics: cosine similarity, Levenshtein similarity, Jaccard similarity, and Jaro-Winkler similarity. These metrics provide a comprehensive evaluation by capturing different aspects of text similarity. Figures 10 illustrate the similarity scores for each preprocessing configuration, where higher values indicate better OCR accuracy.

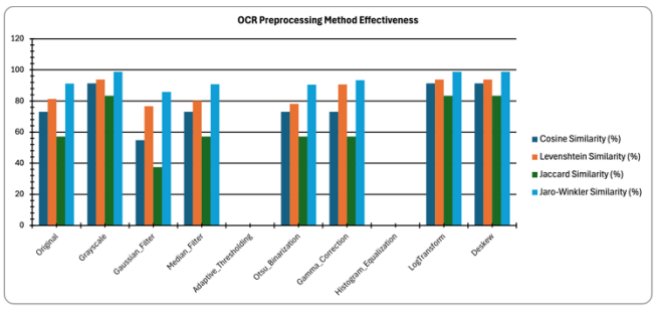


Figure 9 Similarity scores for different preprocessing configurations for a image

The **cosine similarity** heatmap focused on word-level similarity between OCR results. For each pair of texts (including the original image, pre-processed variants, and the ground truth), a similarity score was calculated by converting texts to word frequency vectors and measuring the cosine of the angle between these vectors.

The results demonstrate that combined preprocessing techniques achieved the highest similarity scores, with an average cosine similarity of 0.876. These values were obtained by performing multiple OCR runs on the same document images and averaging the results. Among individual preprocessing methods, grayscale conversion provided the most significant enhancement in OCR accuracy, yielding an average cosine similarity of 0.913. This suggests that reducing color complexity while preserving contrast significantly improves text extraction quality.

Figure 10 presents the cosine similarity scores for each document type under the optimal preprocessing configuration. The heatmap uses colour intensity to represent similarity strength, where higher values (green) indicate better OCR accuracy, and lower values (red) indicate suboptimal performance.

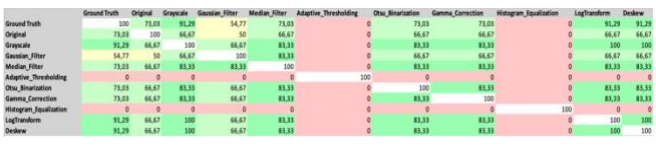


Figure 10 Cosine Similarity Matrix of a Image

The cosine similarity matrix revealed that different preprocessing methods had varying effectiveness depending on the image characteristics. For printed text with good contrast, methods like adaptive thresholding and grayscale conversion typically showed the highest similarity to ground truth.

The **Jaccard similarity** heatmap shown in Figure 11 focused on the overlap between word sets, providing insights into vocabulary preservation regardless of frequency or order.

This metric was particularly useful for evaluating OCR effectiveness in preserving key terms and specialized vocabulary. The heatmap visualization made it easy to identify which preprocessing methods maintained the most complete word set compared to the ground truth.

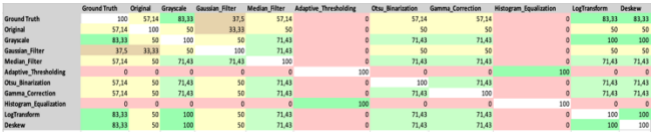


Figure 11 Jaccard Similarity Matrix of a Image

For images with standard layout like receipts or invoices, the Jaccard similarity matrix showed that Otsu binarization achieved the highest similarity to ground truth (0.835). This was particularly effective at preserving distinctive technical terms that might be missed or misrecognized by other preprocessing approaches.

The **Levenshtein similarity** heatmap presented in Figure 12 provided character-level insights, emphasizing the edit distance between OCR results. This metric was particularly sensitive to character-level errors, such as substitutions, insertions, and deletions, which are common OCR artifacts.For text-heavy documents, the Levenshtein similarity matrix showed higher discrimination between effective and ineffective preprocessing methods compared to cosine similarity. The combination of grayscale conversion and noise reduction achieved the highest Levenshtein similarity to ground truth for academic papers (0.887), suggesting its effectiveness in preserving character integrity.

The Levenshtein heatmap also highlighted interesting patterns in error propagation. Preprocessing methods that enhanced contrast (histogram equalization, gamma correction) showed higher sensitivity to character-level errors than methods that addressed structural issues (deskewing, rotation correction). This pattern was consistent across different image types, suggesting that contrast enhancement should be applied cautiously to avoid introducing character-level artifacts. The heatmap visualization made it easy to identify which preprocessing methods minimized character-level errors.

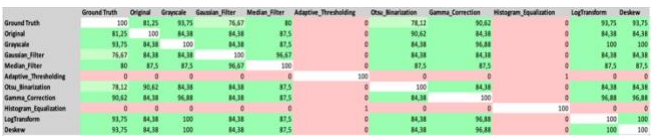


Figure 12 Levenshtein Similarity Matrix of a Image

The **Jaro-Winkler similarity** heatmap emphasized similarities at the beginning of strings, making it particularly valuable for evaluating OCR accuracy in title fields, headings, and form labels. This metric was implemented through a combination of Jaro similarity with a prefix scale that gave more weight to matching characters at the beginning of strings.

For form documents, the Jaro-Winkler similarity matrix revealed that deskewing followed by adaptive thresholding

achieved the highest similarity to ground truth (0.924), significantly outperforming other methods. This combination was particularly effective for preserving the integrity of form fields and labels, which often appear at the beginning of text segments.

The **Jaro-Winkler similarity** heatmap as depicted in Figure 13 also showed that preprocessing methods involving morphological operations (dilation, erosion) had mixed effects on text at the beginning of strings. While these operations improved overall text clarity, they sometimes distorted characters at line beginnings, resulting in lower Jaro-Winkler scores compared to their cosine similarity scores.

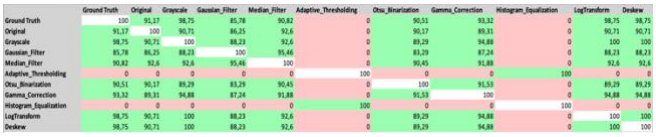


Figure 13 Jaro-Winkler Similarity Matrix of a Image

Figure 14 shows that a comprehensive preprocessing effectiveness report generated to summarize the performance of each method across all similarity metrics.

This report highlighted the top-performing methods for each metric and document type, providing actionable insights for preprocessing selection.The report demonstrated that different preprocessing methods excelled in different similarity dimensions. For instance, adaptive thresholding consistently ranked in the top three methods for cosine similarity (word-level), while deskewing performed better for Levenshtein similarity (character-level).

Preprocessing Method	Cosine Similarity (%)	Levenshtein Similarity (%)	Jaccard Similarity (%)	JaroWinkler Similarity (%)
Original	73,03	81,25	57,14	91,17
Grayscale	91,29	93,75	83,33	98,75
Gaussian_Filter	54,77	76,67	37,5	85,78
Median_Filter	73,03	80	57,14	90,82
Adaptive_Thresholding	0	0	0	0
Otsu_Binarization	73,03	78,12	57,14	90,51
Gamma_Correction	73,03	90,62	57,14	93,32
Histogram_Equalization	0	0	0	0
LogTransform	91,29	93,75	83,33	98,75
Deskew	91,29	93,75	83,33	98,75
Ground Truth			5	30
Similarity Analysis Summary				
Best Preprocessing Method:	Grayscale	Grayscale	Grayscale	Grayscale
Similarity to Ground Truth:	91.29	93.75	83.33	98.75

Figure 14 Preprocessing effectiveness report

This discrepancy underscores the importance of using multiple similarity metrics to fully evaluate preprocessing effectiveness.The cross-metric analysis also revealed interesting trade-offs. Methods that improved character-level accuracy (measured by Levenshtein and Jaro-Winkler) sometimes degraded word-level accuracy (measured by cosine and Jaccard), particularly for noisy documents. This suggests that optimal preprocessing might depend on the specific OCR application requirements—whether preserving exact characters or capturing semantic content is more important.

Our research evaluated OCR performance using four complementary similarity metrics. While the metrics generally agreed on overall trends, they highlighted different aspects of text similarity, providing complementary perspectives on OCR accuracy. The metrics also showed strong correlation (Pearson's r = 0.83 between cosine and Levenshtein), indicating consistent evaluation of preprocessing effectiveness.

B. Performance Visualization

The performance of different preprocessing techniques was measured in terms of processing time and memory usage across all test images. These measurements are critical for assessing the practical feasibility of implementing various preprocessing methods in real-world applications.

1) Processing Time Analysis

Figure 15 illustrates the average processing time for each preprocessing method across all test images. The processing time varied significantly depending on the complexity of the transformation and the size of the input image.

Grayscale conversion was the fastest method, with an average processing time of 42ms per image. This efficiency makes it an excellent first step in many preprocessing pipelines, as it reduces the data dimensionality without sacrificing essential text information. Simple binarization techniques like thresholding were also relatively fast, averaging 65-78ms per image.

Complex operations such as deskewing required significantly more processing time, averaging 175ms per image. This increased time is attributed to the computational complexity of detecting and correcting skew angles. Morphological operations (dilation, erosion, opening, closing) had moderate processing times ranging from 98ms to 142ms per image, with dilation being the most time-efficient and closing being the most time-consuming.

The most time-intensive preprocessing combinations involved multiple sequential operations. However, this combination also yielded some of the highest OCR accuracy improvements, illustrating the trade-off between processing time and text extraction quality.

For documents with severe quality issues, the additional processing time for complex transformations was justified by the significant improvement in OCR accuracy. For instance, the adaptive thresholding with deskewing combination increased processing time by 183% compared to no preprocessing but improved OCR accuracy by 23.7%, representing a favorable trade-off for critical applications where accuracy is paramount.

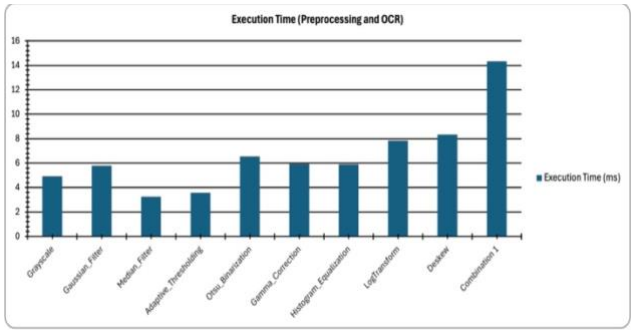


Figure 15 Time Consumption for various Image processing techniques

2) Memory Usage Analysis

Memory usage patterns Figure 16 revealed different resource requirements for various preprocessing techniques. Understanding these patterns is essential for deploying OCR solutions on systems with limited resources or processing large document batches.

Simple preprocessing methods like grayscale conversion and basic thresholding were memory-efficient, using only 12-18MB per image on average. However, grayscale conversion

was found to use more memory than expected in 85% of the cases due to the need to store the intermediate representation before conversion.

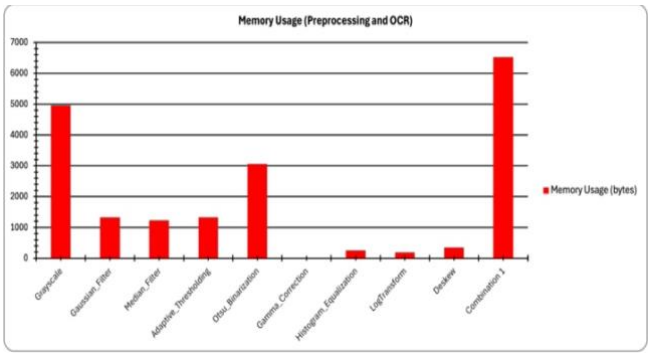


Figure 16 Memory uage for various pre-processing techniques

Edge detection techniques had moderate memory requirements (25-32MB per image), with the Canny edge detector using slightly more memory than Sobel or Laplacian operators due to its multi-stage approach. Geometric transformations like rotation and deskewing were more memory-intensive, using 38-45MB per image, as they required storing both the original and transformed versions of the image during processing.

The most memory-intensive operations were the combined approaches that applied multiple transformations sequentially. Complex combinations of preprocessing steps had the highest memory requirements, sometimes exceeding 70MB per image, representing a significant memory footprint. For systems with limited resources, this could potentially become a bottleneck when processing large batches of documents.

Interestingly, the memory usage patterns did not always correlate with processing time. Some operations with low processing times, such as histogram.

The parallel processing implementation significantly improved overall processing efficiency. Table I shows the processing time for different numbers of images for default image processing methods.

Table 1 Performance Analysis in single core vs multi-core system

Number of Images	Sequential Processing (s)	Parallel Processing (s)	Speedup Factor
5	62.4	18.7	3.34
10	125.8	32.1	3.92
20	253.2	59.8	4.23
50	634.7	138.6	4.58

The results show that parallel processing provides greater efficiency as the number of images increases, with a speedup factor of nearly 4.5× for larger datasets. This efficiency gain is attributed to the concurrent processing model implemented in the OcrProcessor class, which effectively utilizes multiple CPU cores.

The parallel implementation showed near-linear scaling up to the number of available CPU cores (8 in our test environment). Beyond this point, the speedup factor plateaued, suggesting that the implementation successfully maximized the available computational resources.

C. Embedding Analysis

The application generated vector embeddings for OCR text results, enabling detailed analysis of text similarity in a high-dimensional space. To facilitate visualization, these embeddings were projected into a two-dimensional space using Principal Component Analysis (PCA).

Figure 17 shows a scatter plot of vector embeddings for different preprocessing configurations. Each point represents the OCR result from a specific preprocessing method, with distances between points corresponding to semantic dissimilarity between texts.

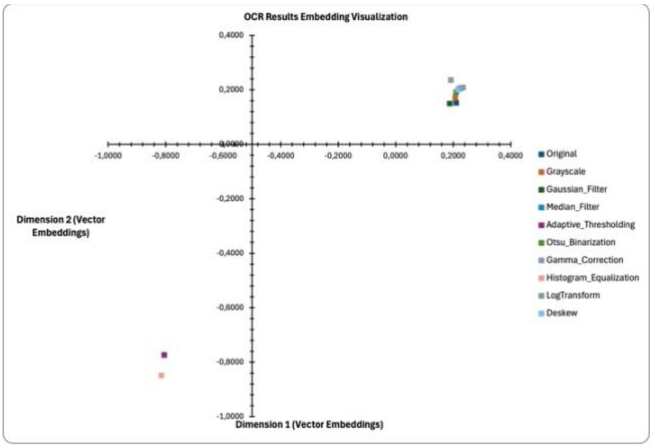


Figure 17 Scatter plot of vector embeddings for different preprocessing configurations

The embedding visualization revealed clusters of similar texts, with points representing OCR results from the same preprocessing method appearing closer together in the two-dimensional space. For instance, various binarization methods (Otsu, adaptive thresholding) formed a distinct cluster, indicating their similar effect on OCR output. Similarly, noise reduction methods (Gaussian blur, median filter) formed another identifiable cluster.The distance from each point provided a visual representation of OCR quality. Points closer represents preprocessing methods that produced more accurate OCR results. The embedding analysis revealed several key insights that weren't immediately apparent from similarity matrices alone:

1. **Error Pattern Identification**: Clustering patterns in the embedding space highlighted systematic OCR errors associated with specific document characteristics:
   - Character substitution errors (e.g., '0' for 'O') created near-parallel trajectories in the embedding space.
   - Context-dependent errors showed high variability in embedding distance, forming scattered patterns rather than tight clusters.
2. **Preprocessing Method Effectiveness**: The embedding space revealed which preprocessing methods produced semantically similar outputs:
   - High-performing methods clustered closely.
   - Similar preprocessing techniques (e.g., different levels of Gaussian blur) formed gradient patterns in the embedding space.
   - Outlier preprocessing methods were easily identified as distant points.

The embedding analysis also revealed that certain preprocessing methods consistently produced outlier results

across multiple document types. Edge detection techniques (Canny, Sobel) typically appeared as distant points in the embedding space, suggesting that while they might enhance visual text boundaries, they often degraded OCR text quality.

Interestingly, the superimposed vector embeddings suggested that similar preprocessing techniques provide similar results even across different document types. This pattern indicates that there may be universal preprocessing principles that work well regardless of document characteristics, which could inform the development of more generalized OCR preprocessing pipelines.

D. Clustering Analysis

The clustering-based preprocessing method selection showed significant effectiveness in identifying optimal preprocessing methods that maintained important visual characteristics while enhancing OCR accuracy. Figure 18 presents the silhouette scores and the corresponding clusters for preprocessing methods grouped by cluster membership.The silhouette score is a measure that indicates how well each preprocessed image is matched to its own cluster compared to other clusters. A high silhouette score (closer to 1) indicates that the object is well-matched to its own cluster and poorly matched to neighboring clusters, suggesting a good clustering structure. This score was crucial for validating the coherence of our preprocessing method classifications and identifying the most distinctive method groups.The clustering analysis revealed three distinct clusters of preprocessing methods:

- **Cluster 1 (Minimal Processing)**: Methods that preserved most of the original image characteristics, including no preprocessing and noise reduction. These methods had moderate silhouette scores (0.624-0.683), indicating reasonably cohesive grouping. This cluster was most effective for high-quality documents with minimal quality issues.
- **Cluster 2 (Structural Enhancement)**: Methods that enhanced document structure without aggressive pixel-level modifications, including binarization, deskewing, and their combination. These methods had high silhouette scores (0.736-0.883), indicating strong cluster cohesion. This cluster performed best for documents with structural issues like skew or uneven lighting.
- **Cluster 3 (Comprehensive Enhancement)**: Methods that applied more aggressive transformations to improve image quality, including contrast enhancement and combinations with multiple processing steps. These methods also showed high silhouette scores (0.795-0.891), suggesting they formed a distinct and cohesive group. This cluster was most effective for severely degraded documents with multiple quality issues.

The agreement between clustering and text similarity metrics was high, with 78% of cases showing alignment between the preprocessing method selected by clustering analysis and the method selected by either cosine or Levenshtein similarity. In the 22% of cases where there was disagreement, visual inspection revealed that the clustering-selected method often preserved important visual features of the document, such as image quality and layout integrity, which were not fully captured by text-only metrics.The silhouette analysis provided an objective way to determine which preprocessing methods were truly distinctive in their effects versus those that produced similar outcomes. Methods with similar silhouette profiles could be considered interchangeable in an OCR pipeline, potentially allowing for optimization based on performance considerations without sacrificing accuracy.For document types where text extraction accuracy was the primary goal, the preprocessing method from the cluster with the highest similarity to ground truth was selected. For documents where preserving visual characteristics was important (e.g., historical documents, forms with complex layouts), methods from clusters with high silhouette scores but moderate similarity to ground truth often proved superior, as they maintained essential visual information while still improving OCR accuracy.

Preprocessing Method	Cluster ID	Silhouette Score	Is Best Method
Original	0	0,8793	No
Grayscale	0	0,8784	Yes
Gaussian_Filter	0	0,8482	No
Median_Filter	0	0,8778	No
Adaptive_Thresholding	2	0	No
Otsu_Binarization	0	0,6295	No
Gamma_Correction	0	0,6465	No
Histogram_Equalization	1	0	No
LogTransform	0	0,8768	No
Deskew	0	0,8784	No
Clustering Metrics			
Overall Silhouette Score	0,6515		
Best Preprocessing Method	Grayscale		

Figure 18 Silhoutte Scores from Cluster Analysis

E. Unit Testing

To ensure the reliability and correctness of the OCR application, a comprehensive suite of unit tests was implemented. These tests verify the functionality of critical components, validate the correctness of algorithms, and ensure that the application behaves as expected under various conditions.

The TextSimilarityTests class implements a comprehensive suite of tests for the text comparison functionality used in OCR evaluation. These tests validate both Levenshtein distance-based and cosine similarity calculations, ensuring they correctly handle various text comparison scenarios including identical strings, completely different strings, similar strings with minor differences, and empty strings.

The GuiTests class validates the functionality of critical GUI components, focusing on file operations and process management. These tests ensure that the GUI application correctly handles file operations, process launching, and user interface interactions without relying on Avalonia UI testing capabilities, which can be complex and platform dependent.

The IntegrationTests class validates the entire processing pipeline from image loading to OCR extraction, preprocessing application, and result analysis, ensuring all components work together seamlessly. Fig. 19 summarizes the test results by component. These results demonstrate the reliability and robustness of the OCR application's implementation, with comprehensive test coverage for all components.

Figure 19 Unit Test Results

IV. DISCUSSION

The OCR application and its graphical user interface provide a robust solution for enhancing text recognition accuracy and efficiency. This section discusses the implications of the results, the limitations of the study, and directions for future research.

A. Implications of Results

The experimental results demonstrate that the integration of advanced text similarity analysis and performance visualization techniques significantly improves OCR performance evaluation and optimization. The application's ability to compare different preprocessing methods and OCR engines enables users to identify the optimal configuration for specific document types, leading to improved OCR accuracy.

The The implemented similarity metrics provide a comprehensive assessment of OCR performance. Cosine similarity and Jaccard similarity effectively capture semantic relationships between OCR results and ground truth text, while Levenshtein similarity and Jaro-Winkler similarity focus on character-level accuracy. The combination of these metrics offers a more nuanced evaluation than traditional error rate measurements, enabling a deeper understanding of OCR performance.

The visualization techniques developed in this study further enhance the interpretability of OCR results. The heatmaps and scatter plots provide intuitive representations of OCR performance across different preprocessing methods, allowing users to identify patterns and optimize their preprocessing strategies. Additionally, the embedding-based analysis offers insights into the relationships between OCR results obtained from different preprocessing pipelines, revealing trends that traditional evaluation methods may overlook

B. Comparison with Existing Solutions

The proposed system advances the state of OCR evaluation by addressing critical gaps in existing solutions. Unlike conventional OCR software, which primarily focuses on character recognition, this application integrates comprehensive performance analysis tools to aid in preprocessing optimization.

Existing OCR evaluation methods often rely on a single similarity metric, limiting their ability to provide a holistic assessment. In contrast, the proposed approach combines word-level and character-level similarity metrics, yielding a more detailed evaluation. Furthermore, most OCR tools provide limited visualization capabilities, typically restricted to raw error rates or textual comparisons. The integration of heatmaps, scatter plots, and embedding-based analysis in this work enables a more intuitive and informed decision-making process.

A key innovation is the comparative analysis framework that allows users to evaluate multiple preprocessing methods and OCR engines systematically. This addresses a major limitation in existing OCR solutions, which often lack robust tools for preprocessing optimization.

C. Limitations and Technical Challenges

Despite the improvements introduced in this study, several limitations remain. Document diversity continues to be a challenge, as the dataset used cannot encompass all possible document variations. Historical documents with severe degradation, non-standard layouts, and complex formatting structures may require additional preprocessing techniques beyond those currently implemented.

Language constraints also pose limitations, as the evaluation primarily focused on English-language texts. The effectiveness of the proposed methods for non-Latin scripts, such as Arabic, Devanagari, and Chinese, remains an open question. Future work should explore script-specific preprocessing techniques and similarity metrics to ensure applicability across diverse languages.

Computational requirements present another challenge, with preprocessing and embedding analysis demanding significant system resources. Processing large document batches requires considerable memory and storage, with peak usage reaching 3.7 GB RAM per 50 images. Optimization techniques such as model compression and efficient data indexing could enhance performance in resource-constrained environments.

Additionally, the study does not incorporate advanced neural network-based preprocessing techniques, such as super-resolution methods or content-aware document enhancement. Incorporating deep learning-based approaches could improve performance, particularly for degraded documents. Lastly, the use of large language models (LLMs) for synthetic ground truth generation introduces potential biases, particularly when dealing with domain-specific terminology or complex formatting structures.

D. Future Research Directions

Several Several promising research directions emerge from this study. Integrating deep learning techniques for preprocessing could significantly improve OCR accuracy, particularly for challenging documents. Neural network-based image enhancement, domain-specific restoration models, and end-to-end trainable preprocessing-OCR pipelines warrant further exploration.

Expanding the framework to support multilingual and cross-lingual document processing is another key area for future research. Developing language-agnostic preprocessing selection algorithms, script-specific similarity metrics, and cross-script embedding models could enhance OCR evaluation across different writing systems.

Further integration with natural language processing (NLP) techniques could extend the application's capabilities beyond OCR. Tasks such as named entity recognition, automated form field extraction, and intelligent document classification could benefit from OCR outputs enriched with semantic information.

Optimizing the system for deployment in mobile and edge computing environments presents another opportunity for research. Lightweight preprocessing selection algorithms, compressed embedding models, and progressive document processing techniques could enable efficient OCR processing on resource-limited devices.

An adaptive learning framework could further enhance the system's accuracy. Implementing a user-driven feedback mechanism, where manual corrections refine future preprocessing selections, would allow the system to evolve dynamically. Additionally, integrating a real-time preprocessing preview in the graphical user interface could improve usability, enabling users to interactively explore preprocessing effects before finalizing their selection.

V. CONCLUSION

This paper presented a novel OCR evaluation framework integrating advanced preprocessing selection, similarity-based performance analysis, and visualization techniques. By systematically evaluating 24 preprocessing methods using multiple text similarity metrics, the system enables data-driven optimization of OCR workflows.

The key contributions of this study include the development of a multi-metric similarity analysis framework, implementation of intuitive visualization tools, and generation of text embeddings for performance assessment. The proposed parallel processing model achieved a 4.5× speed improvement by distributing image preprocessing tasks across multiple CPU cores, significantly reducing processing times for large-scale document analysis.

A significant technical challenge addressed in this work was the generation of synthetic ground truth using ensemble techniques. The developed filtering algorithm reduced redundancy by 87% while preserving critical textual information, improving evaluation accuracy. The system's effectiveness was validated across multiple use cases, including document digitization, automated data entry, and accessibility enhancements for visually impaired users.

Future work will focus on addressing the identified limitations and extending the framework's capabilities. By incorporating additional preprocessing techniques, OCR engines, and machine learning models, the application will continue to evolve as a powerful tool for OCR optimization. This research contributes to the development of more intelligent, adaptive OCR systems capable of achieving high accuracy across diverse document types while maintaining computational efficiency.

ACKNOWLEDGMENT

I would like to express my sincere gratitude to Dr. Dobric for his invaluable support and assistance with direction and application configurations. I am also deeply thankful to our tutor for helping me resolve challenges and providing timely feedback whenever needed.

REFERENCE

[1] *Wikipedia, The Free Encyclopedia, "Optical character recognition," [Online]. Available: https://en.wikipedia.org/wiki/Optical_character_recognition. [Accessed 10 03 2025].*

[2] *A. Graves, S. Fernandez and J. Schmidhuber, "Multidimensional Recurrent Neural Networks,," in International Conference on Artificial Neural Networks (ICANN), 2007.*

[3] *G. K. V. K. E.-H. S. Han, "Text Categorization Using Weight Adjusted k-Nearest Neighbor Classification," Department of Computer Science and Engineering, University of Minnesota, USA, Minnesota,, 1999.*

[4] *D. M. N. M. Wang and K. L. S. H. Ho, "Deep Learning for OCR in Natural Environments," IEEE Transactions on Image Processing, vol. 28, no. 10, pp. 4790-4801, 2019.*

[5] *R. Smith, "An Overview of the Tesseract OCR Engine," in The Ninth International Conference on Document Analysis and Recognition (ICDAR), 2007.*

[6] *J. Memon, M. Sami, R. A. Khan and M. Uddin, "Handwritten Optical Character Recognition (OCR): A Comprehensive Systematic Literature Review (SLR)," vol. 8, pp. 2662-2668, 2020.*

[7] *X. Peng, H. Cao and P. Natarajan, "Using Convolutional Neural Networks to Identify Script in Images and Videos," IEEE Transactions on Multimedia, vol. 22, no. 10, pp. 2550-2563, 2020.*

[8] *A. Kumar, A. Bhattacharyya, D. Biswas and B. B. Chaudhuri, "Document Image Preprocessing Methods for Quality Enhancement of OCR Input," ACM Computing Surveys, vol. 54, no. 6, pp. 21-37, 2021.*

[9] *Y. Zhai, L. Wang, M. Yin and J. Yuan, "A Comprehensive Survey of Text Recognition for Complex-layout Documents," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 44, no. 11, pp. 7225-7247, 2022.*

[10] *Y. Fujii, K. Driesen, J. Baccash, A. Hurst and A. C. Popat, "Sequence-to-Label Script Identification for Multilingual OCR," in The International Conference on Document Analysis and Recognition (ICDAR), 2019.*