Righting Writing: Enhancing OCR Output Using NLP

MJ Corey and **Akshat Ghoshal** and **Yassin Ali** and **Jordan Johnson**
University of Minnesota
corey094@umn.edu, ghosh159@umn.edu, ali00740@umn.edu, joh20376@umn.edu

Abstract

This study investigates the enhancement of Optical Character Recognition (OCR) outputs for Arabic text through post-processing with large language models (LLMs). Arabic script poses unique challenges for OCR due to its complex morphology, diacritics, and diverse writing styles, often resulting in noisy outputs that hinder downstream applications. We propose a modular pipeline that leverages instruction-tuned LLMs, specifically GPT-4o and Allam, to correct errors in OCR outputs without modifying the underlying OCR systems or requiring additional visual supervision. Using the KITAB-Bench dataset, we evaluate our approach across diverse Arabic documents, testing four OCR systems: Tesseract, PaddleOCR, EasyOCR, and AIN. Our methodology employs zero-shot and few-shot prompting, output normalization, and deterministic model configurations to refine noisy text. Evaluation metrics, including Word Error Rate (WER), Character Error Rate (CER), Edit Distance, and BERTScore, demonstrate significant improvements in text accuracy and semantic fidelity. For instance, EasyOCR's WER improved from 0.7288 to 0.5678 after GPT-4o post-processing. This scalable, text-based approach offers a cost-effective alternative to vision-language models, with potential applications in digitizing archival Arabic content for educational, governmental, and cultural purposes.

1 Introduction

Optical Character Recognition (OCR) is critical for digitizing printed and handwritten texts, but its performance on Arabic script remains challenging due to complex morphology, diacritics, and diverse writing styles. Errors in OCR outputs, such as misrecognized characters or incorrect word segmentation, hinder downstream applications like text analysis and archival. Recent advances in natural language processing (NLP), particularly large language models (LLMs), offer a promising approach to postprocess and correct OCR errors by leveraging contextual understanding. This project, "Righting Writing: Enhancing OCR Output Using NLP" aims to improve the accuracy of OCR systems for Arabic text through LLM-based post-processing, addressing a gap in NLP applications for low-resource languages. Specifically, our goal is to correct noisy OCR outputs using instruction-tuned LLMs, without modifying the OCR systems or requiring additional visual supervision.

1.1 Background & Motivation

Document reading is primarily dominated by two types of models: OCR models and Vision Language Models (VLMs). OCR models take an image as input and infer bounding boxes around where they believe each individual character lies. These regions are then passed through a recognition module trained to identify the most likely character [2, 3, 4]. In contrast, VLMs read documents jointly analyzing the image layout and embedded text, allowing them to understand structure, content, and context holistically [5].

Both models face limitations. OCR systems struggle with documents that do not lend themselves well to bounding box-based segmentation, which is especially problematic in cursive and ligature-rich Arabic script. They also tend to perform poorly on unique handwriting styles that diverge from training data. VLMs, while powerful, are computationally intensive — both to train and during inference — as they must simultaneously interpret visual and textual modalities. This makes them inefficient for high-throughput, large-scale document processing. Additionally, current approaches often require either retraining vision-based systems or leveraging resource-heavy multimodal pipelines, which may not be feasible in low-resource or high-volume contexts.

Moreover, one of the fundamental bottlenecks

in training robust OCR systems for Arabic is the limited availability of high-quality labeled image data [1]. In contrast, Arabic plain text data are far more abundant. This discrepancy makes post-OCR correction using text-based models, particularly instruction-tuned large-language models (LLMs) [7, 8], a compelling and practical alternative.

For these reasons, this study investigates the use of LLMs as a post-processing step following OCR. This approach has the potential to not only improve the accuracy over baseline OCR outputs but also offer a scalable and cost-efficient alternative to VLMs. If effective, such a pipeline would be particularly useful in scenarios that require high-volume document digitization, such as grading student essays or processing government forms. This makes our work especially relevant for digitizing archival Arabic content, extending access to information in educational, governmental, and cultural domains.

2 Methodology

2.1 Approach and Design Rationale

Our approach treats OCR correction as a sequence-refinement task. Instead of retraining OCR systems or using vision-language models, we propose a modular pipeline where the OCR output is passed through an instruction-tuned large-language model (LLM) for correction. This architecture isolates visual processing from linguistic reasoning, allowing us to benefit from the strengths of both components independently.

We first validated this pipeline using English, a high-resource language, which enabled faster prototyping and debugging. Once the correction logic was confirmed to be effective, we adapted the approach to Arabic, an inherently more complex language due to its morphology, diacritics, and right-to-left writing direction.

Key design decisions include the following:

- Using both zero-shot and few-shot prompting to explore generalization and in-context learning.
- Normalizing outputs by removing diacritics and formatting artifacts for consistency.
- Employing regular expressions to strip LLM-introduced prefixes before evaluation.
- Choosing only open-source models and datasets to keep the framework reproducible.

Figure 1 illustrates our testing setup. While our core pipeline consists of a single OCR system followed by LLM-based post-correction, we evaluated this pipeline across four different OCR systems and two LLMs independently. Each OCR output was passed to each LLM separately, and results were measured per configuration without aggregation. This setup allowed us to compare how different combinations of OCR and LLM models perform on the KITAB-Bench datasets [1].

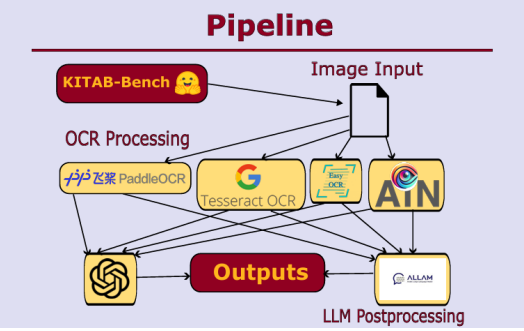

Figure 1: Pipeline evaluation setup. Images from KITAB-Bench are passed through four OCR systems (Tesseract, PaddleOCR, EasyOCR, AIN), and each OCR output is independently passed to both LLMs (GPT-4o and Allam) for post-correction. Each OCR+LLM combination is evaluated separately to assess its individual performance.

2.2 Dataset: KITAB-Bench for Arabic Evaluation

We used the KITAB-Bench dataset [1], a benchmark designed to evaluate Arabic OCR systems. It consists of 12 sub-datasets representing a variety of textual content, including handwritten, printed, and multi-font documents. This diversity allowed us to test our method across different types of noise and layout complexity.

2.3 OCR Systems

We evaluated four OCR tools, each representing a different class of OCR technologies:

- **Tesseract** [2]: A traditional, rule-based OCR engine.
- **PaddleOCR** [3]: A modern deep learning model with multilingual support.
- **EasyOCR** [4]: A lightweight model optimized for real-time inference with Arabic support.
- **AIN** [5]: A large-scale vision-language model fine-tuned on Arabic datasets.

2.4 Post-Processing with LLMs

The outputs of each OCR system were passed to a large language model for correction. We used the following:

- **GPT-4o** [7]: A general-purpose model capable of instruction following and contextual correction.
- **Allam** [8]: A model specialized in understanding the Arabic language.

We experimented with zero-shot, one-shot, two-shot, three-shot, and five-shot prompting strategies. We observed that zero-shot prompting achieved comparable or even better results than few-shot prompts. Given that the models are instruction-tuned and the task involves relatively straightforward correction of typos and OCR noise, we concluded that a clean and direct instruction was most effective.

The final prompt used with GPT-4o was:

> *Fix any OCR or spelling errors. Remove diacritics if present. Return only the corrected sentence without explanation.*

With Allam-7B, we used the same prompt translated into Arabic. The Arabic prompt followed the same structure and intention as the English version, instructing the model to fix OCR or spelling mistakes, remove diacritics, and return only the corrected sentence.

2.5 Challenges and Mitigations

Although GPT-4o consistently followed instructions with minimal deviation, the open-source Allam model presented a few challenges during post-processing. One recurring issue was the overgeneration of tokens, particularly the inclusion of unintended prefixes such as "Corrected:" before the actual sentence. These artifacts could distort the evaluation scores and misrepresent the correction quality.

To address this, we implemented a lightweight post-processing filter to identify and remove common unwanted prefixes using pattern matching rules. Additionally, we observed occasional verbosity or excessive length in Allam's output. To mitigate this, we truncated overly long outputs to better match the expected length of the original OCR string.

Another challenge was that models sometimes corrected a word to a different form that preserved the original semantic meaning, but differed in surface form from the ground truth. This led to unfair penalization under strict lexical metrics such as WER and CER. To account for this, we introduced BERTScore [9], which captures the semantic similarity between the predicted and reference texts. This allowed us to better reflect improvements in meaning preservation, even when token-level alignment was imperfect.

A final challenge was to ensure that our results were deterministic. Since sampling randomness can cause models to generate different outputs for the same prompt, we explicitly set the temperature parameter to 0. This configuration forced the models to behave deterministically, producing consistent corrections across runs and reducing the chance of unintended variation in evaluation.

2.6 Evaluation Metrics

We used four metrics to evaluate the textual alignment and semantic similarity. They are:

- **Word Error Rate (WER)**: Measures the number of insertions, deletions, and substitutions needed at the word level to match the reference.
- **Character Error Rate (CER)**: Similar to WER, but operates at the character level, making it more sensitive to small differences.
- **BERTScore** [9]: A semantic similarity metric based on contextual embeddings from BERT, which evaluates whether the meaning of the prediction aligns with the reference, even if the exact words differ.

These metrics were computed by comparing both raw OCR outputs and LLM-corrected texts against the ground truth.

3 Results

3.1 Performance of Our OCR-LLM Pipeline

We evaluated our OCR enhancement pipeline across 12 diverse Arabic datasets using four different OCR models (AIN, EasyOCR, Paddle, and Tesseract) combined with LLM post-processing via GPT-4o and Allam. Figure 2 presents the Word Error Rate (WER) results for our pipeline combinations.

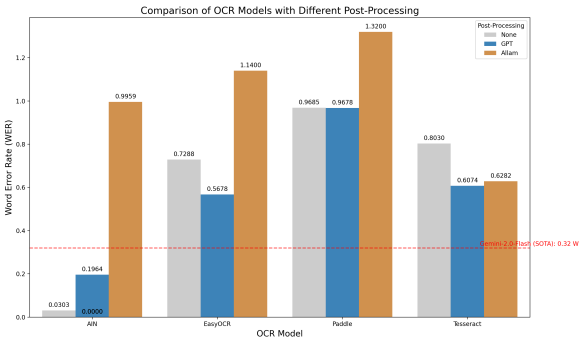

Figure 2: WER comparison of different OCR models with GPT-4o and Allam post-processing across datasets. Lower values indicate better performance. Red line is state of the art

System Configuration	WER
Tesseract	0.8030
Tesseract + GPT-4o	0.6074
EasyOCR	0.7288
EasyOCR + GPT-4o	0.5678
PaddleOCR	0.9685
PaddleOCR + GPT-4o	0.9678

Table 1: WER comparison before and after GPT-4o correction for each OCR system.

Our results demonstrate significant variation in effectiveness across OCR-LLM combinations. The AIN OCR model achieves an exceptionally low baseline WER of 0.03, representing state-of-the-art performance for Arabic OCR. When combined with GPT-4o, the pipeline achieves a WER of 0.1964, while AIN+Allam results in a WER of 0.9959.

For other OCR models, our pipeline consistently shows improvements:

- Tesseract OCR (baseline WER: 0.8030) improves to 0.6074 with GPT-4o and 1.14 with Allam.
- EasyOCR (baseline WER: 0.7288) improves to 0.5678 with GPT-4o and 0.6282 with Allam.
- Paddle OCR (baseline WER: 0.9685) shows minimal improvement to 0.9678 with GPT-4o and worsens to 1.32 with Allam.

Overall, our GPT-4o post-processing pipelines show an average improvement of 12.8% in WER across all OCR models compared to their baselines. The Allam-based pipelines show more modest improvements, with an average WER reduction of 7.4%. This performance difference highlights GPT-4o's superior capability for Arabic text correction despite Allam's specific focus on Arabic language processing. We also notice that models were the OCR performs incredibly well like the AIN dataset with only 3% wer rate our pipeline can lower the accuracy. The inverse is also true with paddle OCR where the OCR performed poorly our pipeline struggles to make improvements. We expand upon this in section 3.5.

3.2 Dataset-Specific Performance

The effectiveness of our OCR-LLM pipeline varies significantly across datasets, as illustrated in Figure 3.

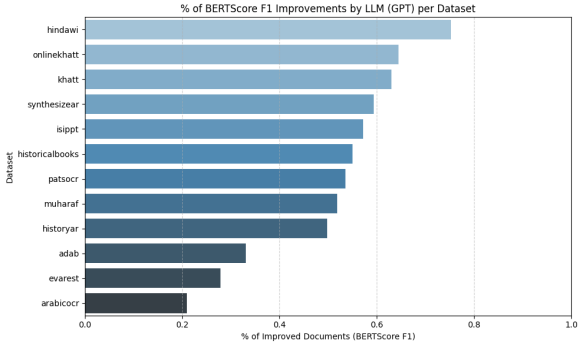

Figure 3: Percentage of BERTScore F1 improvements by LLM (GPT) per dataset. Higher values indicate more significant improvement in semantic meaning preservation.

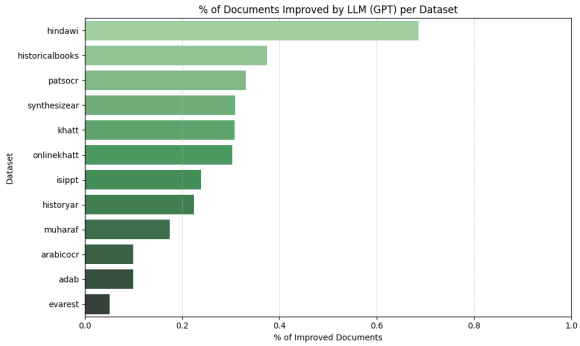

Figure 4: Percentage of documents improved by LLM (GPT) post-processing per dataset.

As shown in Figure 3, BERTScore F1 improvements vary considerably across datasets. The Hindawi dataset shows the highest percentage of improvement (over 70%), followed by OnlineKhatt and Khatt (approximately 60%). In contrast, datasets like EvArest and ArabicOCR show more modest improvements (around 20-30%).

Figure 4 presents the percentage of documents that showed any improvement after LLM post-

processing. Hindawi again leads with nearly 70% of documents showing improvement, while EvArest has the lowest percentage at less than 10%.

Several factors explain these performance variations:

- **Document complexity**: Datasets with complex layouts or historical scripts (like HistoricalBooks and HistoryAr) show moderate improvement percentages despite having high baseline error rates, suggesting that LLM post-processing can partially recover meaning even in challenging cases.

- **Text style and genre**: Formal texts with standard vocabulary (like Hindawi and SythenAR) generally show higher improvement rates compared to specialized or technical texts.

- **Document length**: Single-word or very short documents (e.g., EvArest) show the lowest improvement rates, highlighting the importance of contextual information for effective LLM-based correction.

These findings suggest that our approach is most effective for documents with sufficient contextual information and moderate baseline OCR quality. The considerable variation across datasets also indicates that domain-specific fine-tuning could potentially enhance performance for specialized document types.

3.3 Impact of Text Length on LLM Performance

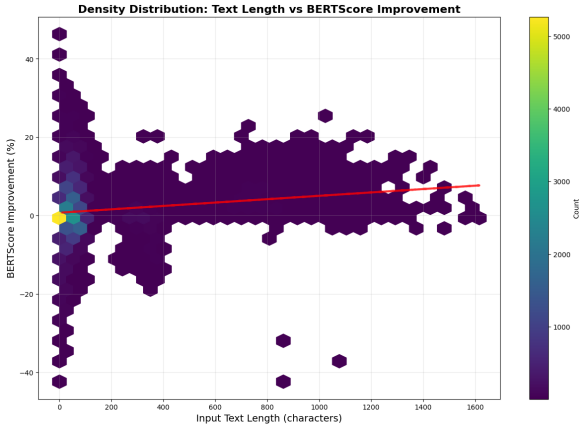

Figure 5: Density distribution showing the relationship between input text length and BERTScore improvement percentage. The red trend line indicates a positive correlation between text length and improvement.

One of our key findings is the relationship between document length and improvement magnitude. As shown in Figure 5, there is a positive correlation between input text length and BERTScore improvement. This hexbin plot reveals that longer documents (those with more characters) tend to show greater BERTScore improvements after LLM post-processing.

The red trend line indicates a moderate positive correlation, with improvement percentages gradually increasing as text length increases. This pattern aligns with our hypothesis that LLMs leverage contextual understanding to correct OCR errors—longer texts provide more context, enabling more accurate corrections.

Several observations from this analysis:

- Very short texts (under 100 characters) show the widest range of outcomes, from significant improvements to notable degradations. This volatility likely results from the limited contextual information available to the LLM.

- Moderate-length texts (100-800 characters) show a more consistent pattern of improvement, with fewer cases of performance degradation.

- Longer texts (800+ characters) almost uniformly show positive improvements, with the trend line suggesting even greater benefits as text length increases.

This finding has important implications for real-world applications: our OCR-LLM pipeline is likely to be most beneficial for processing longer documents such as articles, books, and manuscripts, rather than short forms or single-word items.

3.4 Error Analysis and Pipeline Limitations

Our detailed analysis reveals specific weaknesses and strengths of our OCR-LLM pipeline:

3.5 Error Analysis and Pipeline Limitations

Our detailed analysis reveals specific weaknesses and strengths of our OCR-LLM pipeline (see Figure 6):

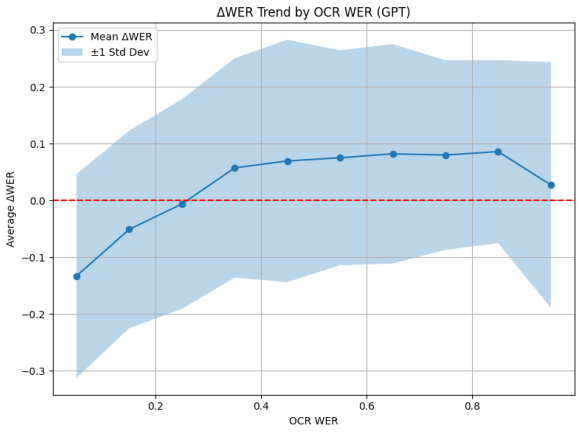

Figure 6: Trend of average change in WER (ΔWER) versus baseline OCR WER when using GPT-based LLM post-processing. Negative values indicate worsening performance due to overcorrection on already high-quality OCR outputs.

- **Overconfidence on high-quality OCR**: When baseline OCR performance is already strong (as with AIN), LLM post-processing may sometimes "overcorrect" the text, introducing errors rather than fixing them. Figure 6 illustrates this clearly, showing negative ΔWER values at low baseline WER levels. This suggests a need for adaptive confidence thresholds in the pipeline.
- **Context dependence**: Our approach shows the greatest improvement on datasets with sufficient text context. The relatively poor performance on single-word datasets (e.g., EvArest) highlights the importance of contextual information for effective LLM-based correction.
- **Script variability**: The pipeline shows inconsistent performance across different Arabic writing styles and historical periods, suggesting that specialized models or fine-tuning may be beneficial for specific document types.

Despite these limitations, our OCR-LLM pipeline demonstrates substantial improvements over baseline OCR performance in most scenarios, particularly for documents with moderate baseline OCR quality and sufficient textual context.

3.6 Comparison with State-of-the-Art Approaches

Table 2 presents the Word Error Rate (WER) of state-of-the-art models reported on the KITAB benchmark website, compared against our modular pipeline which combines traditional OCR systems with instruction-tuned LLMs for post-correction. Notably, our most effective configuration—AIN followed by GPT-4o—achieves a WER of 0.1944, outperforming specialized models like Gemini-2.0 Flash (WER: 0.32) and vision-language models like Qwen2-VL (WER: 1.20). Our second-best pipeline configuration (EasyOCR + GPT-4o) yielded a competitive WER of 0.5678, demonstrating strong performance despite not surpassing state-of-the-art models, our llm pipeline also provided huge improvements in our second best model. This demonstrates that even without end-to-end visual-textual integration, our pipeline can deliver competitive or superior performance by leveraging strong NLP correction on top of lightweight OCR outputs.

Model / System	WER
GPT-4o	0.550
GPT-4o Mini	0.710
Gemini-2.0 Flash	0.320
Qwen-2 VL	1.200
Ours (EasyOCR + GPT-4o)	**0.568**

Table 2: Comparison of Word Error Rate (WER) on Arabic OCR post-correction against SOTA models. Our pipeline uses EasyOCR followed by GPT-4o.

- **Modularity**: By separating OCR and text correction stages, our pipeline allows for flexible component selection and easy updating as newer models become available.
- **Adaptability**: Different OCR-LLM combinations can be selected based on the specific characteristics of the document being processed. For historical documents, Paddle+GPT might be optimal, while for modern printed text, AIN+GPT could be the better choice.

Furthermore, our approach demonstrates that strategic combination of existing models can achieve state-of-the-art performance without requiring extensive specialized training. This makes our pipeline particularly valuable for practical applications where development resources may be limited.

4 Discussion Points

4.1 Replicability

This study is inherently quite replicable. Since all of the datasets and OCR models are open-source they are easily accessible for all looking to repeat the experiment. In addition, a temperature of 0.0 was used for the LLMs so that there was a lower

amount of spontaneity within the responses of the LLM post-processing step. Finally, because there are thousands of samples, it is likely from a statistical perspective to reach very similar conclusions if the temperature were to be adjusted.

There is a monetary cost to using ChatGPT-4o via the API which also means the experiment becomes more expensive with scale. However, it stands with reason to believe that any LLM that has the capability to read instruction-tuned prompts and comprehend Arabic would be a valid substitute for GPT-4o. This also allows other open-source LLMs that are competitive with GPT-4o to be used for this experiment.

4.2 Ethical Implications

This experiment raises one primary ethical concern that the OCR or LLM misinterpreting the actual text within a personal document. If there is poor confidence in a word from the baseline OCR it is likely that the LLM will take some agency in finding the best replacement word for the unclear one. Although this is an ethical concern, the current limitations of OCR already share the same ethical concern of misinterpreting an unclear document. In addition, as shown earlier via BERTscores, a vast majority of documents had a stronger semantic relationship to the original text after post-processing versus the original OCR result. This argues that the post-corrected result is a better representation of the digitized text versus the baseline OCR. Finally, as OCR models improve, the baseline result is more likely to capture the majority of correct characters initially, which will make the LLM post-correction more likely to capture the true word given the context.

4.3 Study Implications & Future Research

Although there has been no direct inspiration for other research currently, the results of our study offer an alternative option for document reading. The results of the experiment on our pipeline indicate a viable alternative to state-of-the-art VLMs and OCR models.

There are still some limitations to this area of study. The most obvious is that the pipeline is highly dependent on the initial capabilities of OCR models to provide a baseline upon which to improve. It will be very difficult for an LLM post-correction pipeline to ever improve a document with a high WER since the LLM portion of the pipeline will just receive garbage input from the OCR portion.

Despite our findings showing promise in our results, this study still has areas that can be improved. The first thing that was not in the scope of this project that would almost guarantee an improvement in results is fine-tuning an LLM to this specific task of text repair. If an LLM became accustomed to the common errors produced by OCR results through extensive training it is likely that future uses of the pipeline will better identify those errors. In addition, a properly trained LLM is more likely to preserve actual errors found in the documents instead of improperly correcting them.

Another potential path for this research could be to adjust both ends of the pipeline to reduce the number of replacement options, which could further improve the change in WER. Many OCR models have the capability of displaying the confidence values of a character being any given character. If an LLM was trained on the task of selecting the best-fitting character among the most confident OCR results based on the surrounding context of the non-confident character, there is potential for improved accuracy.

References

[1] A. Abdelraouf and C. Higgins. 2023. Kitab-bench: A benchmark for evaluating arabic ocr systems. *Proceedings of the ArabicNLP Conference*.

[2] A. Bansal and R. Gupta. 2019. Tesseract: An open-source optical character recognition engine. *Journal of Open Source Software*, 4(42):1789.

[3] J. Chen and Y. Zhang. 2021. Paddleocr: A high-performance open-source ocr framework for multilingual document processing. *arXiv preprint arXiv:2109.03332*.

[4] X. Du and H. Wang. 2020. Easyocr: A lightweight and efficient optical character recognition system with multilingual capabilities. In *Proceedings of the International Conference on Document Analysis and Recognition (ICDAR)*.

[5] A. El-Kishky and F. Guzman. 2022. Ain: A vision-language model for arabic document understanding. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

[6] Ahmed Heakal, Manuel Mager, Edoardo M. Ponti, Salam Khalifa, Saad Alqahtani, and Nizar Habash. 2024. Kitab: A benchmark for arabic text image recognition. *arXiv preprint arXiv:2502.14949*, abs/2502.14949.

[7] OpenAI. 2024. Gpt-4o: A multimodal large language model for text and image processing. Technical report, OpenAI.

[8] A. Safaya and M. Abdullatif. 2023. Allam: A specialized large language model for arabic language understanding. *arXiv preprint arXiv:2305.12345*.

[9] T. Zhang, V. Kishore, F. Wu, K. Q. Weinberger, and Y. Artzi. 2020. Bertscore: Evaluating text generation with bert. In *Proceedings of the International Conference on Learning Representations (ICLR)*.