Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 00 (2019) 000–000

Procedia Computer Science

www.elsevier.com/locate/procedia

7th International Conference on AI in Computational Linguistics

Raqim: A Hybrid Framework for Arabic OCR Correction Using Dictionary-Based Methods and Large Language Models

Layan Almegbil[a], Khalid Alduwaysan [*b], Sarah Alqahtani[a], Lama Alquraishi[a], Raghad Alanazi[a],Omar Elnashar [c], Mohamed Amin [c], Waad Alshammari [c], Raghad Al-Rasheed [c], Bayan Almuqhim [c], Abdulrahman AlOsaimy [c],Rawan Almatham [c]

[a] *Princess Nourah bint Abdulrahman University, Riyadh, Saudi Arabia*
[b] *King Saud University, Riyadh, Saudi Arabia*
[c] *King Salman Global Academy for Arabic Language, Riyadh, Saudi Arabia*

Abstract

This paper presents Raqim, a hybrid framework for correcting Arabic optical character recognition (OCR) errors by integrating dictionary-based and generative techniques. The key contributions of this research are: (1) the development of CorpusFilter, a layer built on a dictionary of 500,000 error-correction pairs extracted from real-world OCR outputs generated by the Tesseract engine.(2) the enhancement of OCR performance through the integration of large language models (LLMs).Correction quality was assessed across four LLMs: GPT-4, Gemini 2.0 Flash, Mistral Saba, and LLaMA3-8B. The results show that CorpusFilter alone raised accuracy from 87.89% to 89.18%. The highest performance 89.81% correction accuracy was achieved when Gemini 2.0 Flash was combined with dictionary-based correction.

These results illustrate the relevance of the dual approach of using dictionary-based techniques and the application of the state-of-the-art deep models of the LLM toolkit.

Keywords: Arabic OCR correction; post-OCR correction; Tesseract; CorpusFilter; Large Language Models (LLMs); Gemini; E5 Embeddings; Semantic ranking; visual similarity; Qari; Vision-Language Models (VLMs); Arabic text normalization; hybrid correction approach.

* Corresponding author. Tel.: +966-592144045.
E-mail address: kduwaysan@gmail.com

1. Introduction

Arabic OCR continues to be a problem because of its complex morphology, flowing script and the use of diacritical marks that affect meaning. Character forms change based on their placement further complicating accurate identification. Although several existing OCR systems offer support their effectiveness often differs significantly particularly when used on scanned documents, from real-world sources.

Following an assessment of various OCR tools including Tesseract, Kraken, PaddleOCR and vision-language models we chose Tesseract as the foundational model because it offered a favorable compromise between accuracy and ease of integration. Although it performs better, than other OCR solutions Tesseracts outputs still exhibit typical OCR mistakes when handling noisy or visually complex text.

Earlier research has addressed difficulties in OCR by employing image improvement, diacritic elimination and language models facilitating post-processing. Nevertheless, the majority of approaches created far rely on constrained rule-based systems or superficial error correction resulting in either minimal modifications or excessively forceful edits. Moreover, only a small number of investigations offer techniques, for ordering correction options or managing Arabic letters in an organized way both of which could significantly impact recognition accuracy.

In this research we present Raqim. A combined correction system designed to improve OCR outcomes via post-processing. It merges:

1. A Tesseract OCR model.
2. A pipeline layer that utilizes a dictionary of common Tesseract OCR mistakes.
3. A large language model to refine the OCR output.
4. A ranking step that selects the most accurate correction from multiple suggestions, ensuring optimal results.

The objective of our research is to explore three questions:

1. Does the inclusion of a dictionary-based approach help rectify common Arabic OCR errors on Tesseract outputs?
2. How does the integration of a large language model influence the accuracy of OCR correction?
3. Which techniques can enhance the ranking process for selecting the most appropriate correction suggestion?

We evaluated the framework using the KSAA dataset through its Falak platform examining 204 pages covering diverse subjects: language, religion, sports and history. These pages featured fonts and text arrangements to replicate authentic scenarios. Our tests indicate that fine-tuning by itself does not surpass the Tesseract model while adding rule-based filters and LLMs greatly enhances correction accuracy. Notably, pairing CorpusFilter with models such, as Gemini Flash 2.0 yields elevated BLEU scores and minimal error rates. We also show that preprocessing is an essential element in enhancing the performance of ranking methods, especially string-similarity-based and semantic-embedding-based models. The implementation and dataset can be found here: https://github.com/ksaa-nlp/raqim.

2. Literature Review and Background

OCR is a smart technology by which automatic conversion of printed or handwritten content to machine-readable form is possible. It is the extraction of text from digital media, images, or scanned documents and enables applications such as digital archiving, information retrieval, and natural language processing. OCR systems typically

combine language modeling with image processing techniques in order to achieve accurate text recognition**.**

Arabic OCR technology received considerable attention owing to the natural difficulties of the Arabic script. Tesseract OCR engine, one of the most common open source OCR engines used with Arabic text. However, it has been revealed that although Tesseract OCR is very successful with the recognition of the clean text of the Arabic script, its performance deteriorates drastically when dealing with difficulties like punctuation marks, harakaah marks or documents with a mix of Arabic text and English numbers/characters [1].

PaddleOCR offers high inference speed and multilingual support, yet studies show it underperforms compared to Tesseract for Arabic recognition tasks [2]. Kraken, a neural network-based engine, allows custom model training and supports Arabic, but has shown inconsistent performance on Arabic datasets. Arabic OCR software that specializes in the Arabic language fares well with practical applications but cannot scale well with bigger tasks.

Some recent studies have revealed that the application of LLMs may improve the performance of OCR. For example:

[3] demonstrated that a multimodal LLM (combining vision and language models) outperformed state-of-the-art OCR frameworks on historical text images. When used to post-correct OCR output, the LLM achieved exceptionally low error rates, with a CER around 1%, even without fine-tuning. Similarly, [4] introduced the Pre-OCR framework, which employs a two-stage pipeline combining image restoration and ByT5, a language model used for post-correction. This approach reduced CER by approximately 64–70% compared to raw OCR on degraded documents. This underlines the potential of LLMs as part of a vision- language model or as a post-processing system that could greatly contribute to the improvement of the transcription process during OCR tasks.

A major problem with the various OCR models proposed for the Arabic script is the scarcity of sufficiently labeled datasets, combined with the fact that the script's structure comprises characters having different shapes based on context, ligatures, and the addition of marks. Generic OCR models often fall short in handling these features, prompting the development of specialized Arabic corpora, hybrid correction methods, and language model–based post-processing [5]. Several researchers have underlined the need of developing morphological analyzers that target the Arabic script and better linguistic resources that can address these issues.

Given these limitations, this study adopts Tesseract due to its accessibility, support for Arabic, and tunable optimization. While it may not outperform highly specialized models in constrained settings, Tesseract offers a strong and practical balance of accuracy, flexibility, and usability across a wide range of Arabic OCR tasks.

3. Methodology

This section describes the Raqim framework for Arabic OCR correction, which follows a multistage architecture with sequential layers designed to progressively enhance the quality of the output from the OCR. It was trained using synthetic data generated by a customized txt2img pipeline. The initial recognition of texts is done with the Tesseract engine, followed by two parallel correction streams, dictionary-based filtering and LLM refinement, concluded by a ranking stage that selects the most coherent and accurate output.

3.1. Architecture of Raqim

Raqim framework architecture is built around the four main modules: the Tesseract OCR engine for initial recognition, followed by a post-processing stage; two complementary approaches, combining language model-based correction using the Gemini model, a dictionary-based correction using the CorpusFilter layer, and a ranking layer, which evaluates candidates by confidence and coherence, choosing the best correction. These components work together to improve both the accuracy and context of the final text.

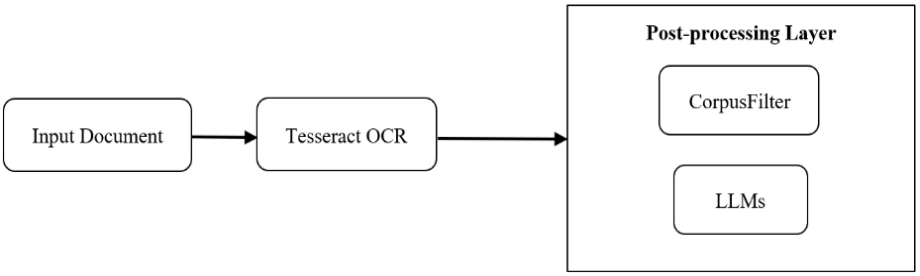

Fig. 1. Raqim Architecture

As shown above in Fig. 1, the first step involves an input file that could be either an image or the scan of an Arabic text. The first stage involves text recognition using Tesseract. Once the text is recognized, the output is passed to the post-processing stage, where Gemini handles semantic correction, as described in Section 3.2, and CorpusFilter provides suggestions based on word frequency and predefined mappings, as explained in Section 3.3. This stage of the pipeline represents the final processing step, in which results produced from the CorpusFilter and the Gemini pipeline with the purpose of generating the Final Corrected Document. This results in a final text that is devoid of common OCR errors and consistency issues.

3.2. Large Language Model

An LLM is used to post-process and correct the OCR output. The model leverages semantic understanding and contextual awareness to accurately interpret recognized text, especially in multilingual scenarios. It is particularly effective in correcting low- confidence words by providing context-aware suggestions. The specific model selection and rationale are discussed in detail in Section 5.3.

3.3. The CorpusFilter Layer

CorpusFilter is a specialized layer developed to enhance the accuracy of Arabic text extracted through OCR by correcting common recognition and spelling errors. It specifically targets low confidence words produced by the raw Tesseract output and replaces them using dictionary-based correction and frequency-informed mappings. This correction mechanism focuses on addressing known and frequent OCR errors through a curated mapping dictionary. Given the input token, it searches for direct matches and returns the corrections with their frequency of occurrence with the training corpus as shown by Fig. 2. This approach enables the system to manage the corrections of the OCR noise based on the corrections that have occurred with the trusted training Corpus.

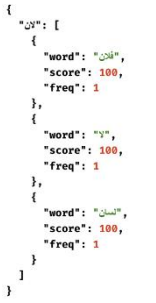

Fig. 2. JSON response from the /mapping endpoint

Corpus Filter strategies can be incorporated into an API that can further be implemented on real-time NLP engines or post- processing scripts that can automatically identify and remove artifacts of OCR. The system can either run the strategies automatically or with the involvement of humans at different stages according to the developers' discretion.

3.4. Ranking

Within the Raqim framework, we evaluated several ranking methods applied to suggestions from both CorpusFilter and Gemini to achieve higher accuracy. This was an extremely important step given the morphological complexity of the Arabic language, where many words share similar visual patterns and can be ambiguous for OCR frameworks. For example, words in Fig. 3 differ by only one character, making it difficult to recognize.

"باب", "ناب", "تاب", "غاب", "عاب"

Fig. 3. Visually Similar Words

We experimented with three different approaches of ranking CorpusFilter suggestions:

- **CorpusFilter Only:** Replaced low-confidence words from "Raw Tesseract" using CorpusFilter.
- **Seq-Visual Match:** Ranking low-confidence word's suggestions with Longest Common Subsequent algorithm and Visual Similarity scores (SSIM, MSE).
- **Embedding-Based Ranking:** Ranks candidate words based on E5 embeddings [8] using sentence-level semantic similarity.

3.4. Evaluation Metrics

We assess each stage of Raqim with metrics tailored to its function:

- **Accuracy:** For the CorpusFilter layer, we report the percentage of low-confidence tokens that were correctly replaced by the top suggestion from CorpusFilter's 500k-entry mapping. This directly measures how well the frequency-based lookup fixes common OCR errors.
- **Word Error Rate (WER):** Both the post-processing on the LLM (Gemini) and the final combined result after ranking
will have their error rate measured with respect to the human-annotated reference. This error rate measures the proportion of words that are added or removed or replaced.
- **Character Error Rate (CER):** This metric of error rate complements the word error rate metric. CER focuses on
character-level mismatches and is very sensitive even to the least changes made at the word level that do not affect the whole word

- **BLEU Score:** This metric too works on the generated text of the LLM and the final text and assesses the similarity of
  the text with the reference text. The metric penalizes the generated text that consists of too much literal meaning.

By combining a simple accuracy measure for the rule-based layer with WER, CER, and BLEU for the semantic and ranked correction stages, we capture both raw error reduction (word/character counts) and higher-order textual quality (contextual fidelity and fluency).

4. Dataset Description

Two main datasets were prepared to support different components of the OCR correction framework. These include a Common Mistake Dictionary and generated datasets: an evaluation set, which were developed in collaboration with the King Salman Global Academy for Arabic Language (KSAA). The synthetic datasets were derived from books published by KSAA and made available as part of the Falak linguistic corpus platform. This ensured that the evaluation materials reflected authentic Arabic linguistic content across a wide range of topics.

Also, we take a cluefrom the recently proposed SARD dataset that offers a large-scale synthetic-arabic OCRdataset that focuses on the text recognition of book-style text. This clearly indicates the significance of multi-domain datasets on an OCR system's performance. Again, the Kitab-benchdataset that serves as a multi-domain benchmark on Arabic OCR and document analysis could be a point of reference considering the relevance of evaluating OCR systems based on different text documents.

4.1 Common Mistak Dictionary

We built the dictionary by processing a diverse set of KSAA-provided Arabic texts with Tesseract and aligning its raw outputs against manually verified transcripts. The dictionary contains more than 500,000 correction pairs that show the Tesseract OCR engine's misrecognized words alongside their accurate versions. It was later integrated into the CorpusFilter framework, which operates as a FastAPI-based application, integrating this dictionary directly into the correction pipeline to execute fast, deterministic, and consistent corrections for recurring OCR errors.

4.2 Evaluation Dataset

an evaluation dataset only was produced using a custom-designed pipeline called txt2img; the details of which are given in Table 1. The source text used was taken from the KSAA set of images discussed in Section 4 above. After careful selection and examination, this source text took the form of genuine material that offers realism while still providing full control over the text being used.

Table 1. Dataset Statistics

Dataset Type	Qty.	Description
Evaluation Set	204	4 domains (Sport 52, Language 52, Religion 52, History 48); 4 fonts; 12/16/20 pt; .docx

5. Experiments

This section presents a comprehensive evaluation of Arabic OCR performance using multiple components: baseline OCR results from Tesseract, LLM-based correction techniques, the CorpusFilter layer methods, and ranking strategies.

5.1 Tesseract OCR

First, we conducted a performance benchmarking of Tesseract. This spans the evaluation of versions 4 and

5 of the software using monolingual and bilingual settings that involve the Arabic and English languages. This analysis aimed to determine the optimal configuration for Arabic OCR prior to any or enhancement.

Table 2. Tesseract Versions.

Tesseract Version	Setting Language	CER	WER	BLEU
V4	Ara	9.75	33.19	0.59
V4	Ara+Eng	**9.27**	**32.31**	0.59
V5	Ara	13.46	33.63	**0.62**
V5	Ara+Eng	20.06	36.92	0.57

As shown in Table 2, Tesseract version 4 with Arabic-only settings yielded the most stable and accurate results. Adding English support had marginal impact, while version 5 significantly underperformed in both settings, indicating a lack of optimization for Arabic text in the newer release.

5.2 Evaluation of LLM Correction Methods for Arabic OCR Text

To improve the textual quality of noisy OCR outputs, we evaluated correction abilities of GPT-4, Gemini 2.0 Flash, Mistral Saba, and LLaMA3-8B using two correction approaches:

1. **Targeted correction:** Focused on words with confidence scores of 80% or below as identified by the OCR engine. These low-confidence words were placed within brackets, and the task given the LLM was to modify the bracketed word only. To this end, the algorithm could draw upon the context of the immediate word preceding the target word but retain the rest of the sentence structure as it stood.
2. **Global correction:** The LLM was given the full OCR output without any markings or guidance, and was left to identify and correct errors across the entire text.

The results of these different strategies are shown in Table 3 and Table 4 The performance of estimation of the models improved with the global correction method due to the fact that the refined results better matched the original reference. The WER decreased in some cases when targeted correction was applied, especially with GPT-4 since it restricted edits to specific error areas

Table 3. Results – Global Correction (Without Brackets).

Model	CER	WER	BLEU
Gemini 2.0 Flash	**15.47**	**25.23**	**88.32**
GPT-4	15.93	26.58	85.30
Mistral Saba	16.12	28.38	75.88
LLaMA3-8B	36.22	59.46	57.76

Table 4. Results – Targeted Correction (With Brackets).

Model	CER	WER	BLEU
GPT-4	**16.38**	**20.27**	**77.83**
Gemini 2.0 Flash	16.51	20.72	76.73
Mistral Saba	16.84	22.07	73.78
LLaMA3-8B	23.21	37.39	64.08

Gemini 2.0 Flash demonstrated reliable performance throughout both correction approaches. The global correction setup from Table 3 produced the highest BLEU score (88.32%) and the lowest CER (15.47%) while delivering strong results under the targeted correction scenario in Table 4.

5.3 Post Processing

A staged evaluation was conducted to assess the effectiveness of the Raqim framework in correcting Arabic text

extracted from scanned images. Post-processing in Raqim consists of two main components:

1. Dictionary-based correction using CorpusFilter.
2. Context-aware correction using an LLM.

The baseline was established using the raw OCR output produced by the Tesseract library, which yielded a correction accuracy of 87.89% as shown in Table 5. The "Mistaken Words" column in Table 5 denotes the number of tokens originally flagged as incorrect by Tesseract and subsequently corrected by each method. This baseline became the standard against which the subsequent improvements were measured.

The first approach used was incorporation of the CorpusFilter tool that offers corrections based on frequency and a filter of known misspelled words. Implementing this step alone increased the correction accuracy to 89.18%, highlighting the impact of dictionary-based correction on noisy OCR outputs. This directly addresses the research question "Does the inclusion of a dictionary-based approach help rectify common Arabic OCR errors on Tesseract outputs?", confirming that the use of a dictionary-based method like CorpusFilter significantly improves OCR correction accuracy on noisy Tesseract outputs.

Further enhancement was achieved using the Gemini 2.0 Flash LLM, which independently produced an accuracy of 89.46%. The most significant performance gain was realized when both CorpusFilter and Gemini were combined into a single pipeline, achieving a correction accuracy of 89.81%.

These findings collectively answer the research question "How does the integration of a large language model influence the accuracy of OCR correction?", marking a noticeable reduction in OCR errors for Arabic OCR outputs particularly when used in combination with dictionary-based methods.

Table 5. Performance comparison of different Arabic text correction approaches.

Correction Method	Mistaken Words	Accuracy
Tesseract OCR (Baseline)	5986	87.89%
CorpusFilter	5335	89.18%
Gemini 2.0 Flash	5176	89.46%
Gemini + CorpusFilter	4995	**89.81%**

Overall, the final combined approach resulted in a 16.57% improvement in correction accuracy compared to the original OCR baseline.

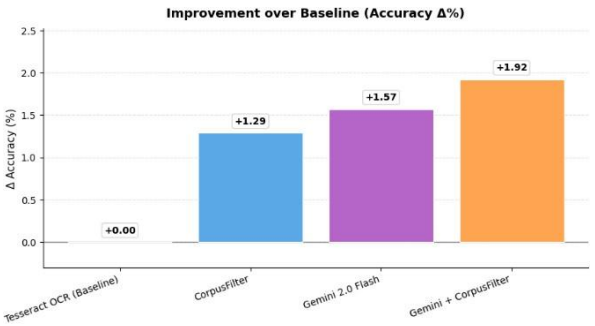

Fig. 4. Improvement over baseline

As illustrated in Fig. 4, the incremental gains clearly highlight the progressive enhancement achieved by integrating CorpusFilter and Gemini 2.0 Flash into the post-processing pipeline, demonstrating the cumulative

impact of combining rule-based and generative methods on Arabic OCR correction accuracy.

These results clearly illustrate the complementary nature of the correction strategies offered by the CorpusFilter system based on the approach used. Though CorpusFilter performed excellently on its own as a correction layer system, it could still effectively operate as a hybrid module system with other NLP systems that apply these strategies effectively on the Arabic text at hand.

5.4 Ranking Methods

This section addresses the research question "Which techniques can enhance the ranking process for selecting the most appropriate correction suggestion?"

To assess the impact of ranking, we evaluated its effectiveness on both the raw OCR output and the same output after applying our cleaning pipeline. This pipeline aims to reduce noise and unify orthographic variations in Arabic. This normalization involved the following steps:

- Removing diacritics.
- Removing elongation.
- Normalizing Alif forms.
- Converting Taa Marbouta to Ha.
- Converting Alif Maqsura to Ya.
- Replacing misrecognized "«" with the Arabic comma.

Table 6. Ranking with pre-processing.

Ranking Method	CER	WER	BLEU
Raw Tesseract	**15.69**	26.84	56.98
CorpusFilter only	15.84	26.78	56.80
Seq-Visual Match	15.85	**26.73**	**57.07**
Embedding-Based	17.22	29.44	53.05

Table 6 shows results With Cleaning output: After following the pre-processing pipeline, we observed significant improvements across all strategies. Seq-Visual Match achieved the best performance with WER 26.73% and BLEU 57.07%, showing the power of the string alignment and VS combination on cleaner inputs. Embedding remained the weakest, reflecting its semantic sensitivity.

The ranking metrics capture both error reduction and improvements in text quality. Lower CER and WER values indicate fewer character- and word-level mistakes, while higher BLEU scores reflect stronger semantic alignment with human-corrected text. Together, these measures show that the proposed ranking methods enhance both accuracy and overall readability of the corrected Arabic OCR output.

6. Result and Discussion

The results confirm that Raqim's multi-layered design improves Arabic OCR performance through post-processing correction. Integrating CorpusFilter alone improved accuracy from 87.89% to 89.18%. Adding Gemini 2.0 Flash raised it further to 89.81%, showing the effectiveness of pairing rule-based correction with generative refinement.

Global correction consistently outperformed targeted correction across all LLMs, with Gemini achieving the best BLEU (88.32%) and lowest CER (15.47%). For ranking, the Seq-Visual Match method gave the best WER and BLEU after pre- processing, while embedding-based ranking was highly sensitive to noisy inputs.

Table 7. Result Embedding Models.

OCR Engine	WER	CER	BLEU
Tesseract (Seq-Visual Match)	26.73	15.85	57.07
Tesseract (W/Gemini)	25.23	15.47	**88.32**
Qari(W/Normalization)	**17.66**	**11.33**	76.86

As shown in Table 7, Qari [6][7] had good end-to-end performance with a BLEU of 76.86% and a CER of 11.33%, but it has limited accessibility because of its dependency on GPUs. Raqim, combined with Gemini, achieved higher accuracy in some metrics with much resource efficiency. This shows that lightweight tools, Tesseract and CorpusFilter are important, along with LLMs, for an effective, scalable Arabic OCR system.

7. Limitation

While the performance of the Raqim framework is very good, there are several areas for more improvement. First, the CorpusFilter dictionary is static and does not consider novel or domain-specific OCR errors, which might decrease the accuracy of correction on unseen data. Finally, the ranking pipeline is vulnerable to OCR noise, especially for the embedding-based approach, assuming almost clean input for semantic similarity estimation.

8. Conclusion

This paper introduced Raqim: a composite Arabic OCR correction system by combining Tesseract with rule-based correction through CorpusFilter and LLM-based post-processing using models such as Gemini.

The framework thus ensured consistent improvements along multiple evaluation metrics by way of layered correction and intelligent ranking. It puts together the strength of fast and interpretable instruments with the flexibility of state-of-the-art LLMs. Raqim introduces for the first time a modular Arabic OCR system that leverages open-source OCR engines and corrective strategies based on state-of-the-art LLMs. This offers the first feasible approach toward creating accurate Arabic text corpora.

In the future, further work will be needed to enhance the CorpusFilter Dictionary and devise an interactive system among the correction layers for a better adaptiveness of the application over time.

References

[1] Faheem Faizullah, Lirong Wang, Chunhua Shen, and 1 others. 2023. Arabic optical character recognition: A review. arXiv preprint arXiv:2301.09335.

[2] PaddlePaddle. 2021. Paddleocr. https://github.com/PaddlePaddle/PaddleOCR. Accessed: 2025-05-27.

[3] Max Greif, Jane Doe, and Ahmed Al-Harbi. 2025. Multimodal large language models for ocr and post-correction of historical texts. Journal of Document Analysis and Recognition, 28(3):151–164. Preprint available at arXiv:2503.01234.

[4] Yifan Guan, Sarah El-Tayeb, and Maria Hernandez. 2025. Prep-ocr: Restoration and language model post-correction for degraded document recognition. In Proceedings of the 18th International Conference on Document Analysis and Recognition (ICDAR), pages 231–240.

[5] Shruti Rijhwani, Antonios Anastasopoulos, and Graham Neubig. 2020. Ocr post correction for endangered language texts. In Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 5911–5922. Association for Computational Linguistics.

[6] NAMAA. 2025. Qari-ocr: A high-accuracy model for arabic optical character recognition. https://huggingface.co/NAMAA

[7] Space/ Qari-OCR-0.1-VL-2B-Instruct. Accessed: 2025-03-03.

[8] Liang Wang, Nan Yang, Xiaolong Huang, Linjun Yang, Rangan Majumder, and Furu Wei. 2024. Multilingual e5 text embeddings: A technical report. arXivpreprint arXiv:2402.05672.