

Indic Document Intelligence: Recognition, Correction, Translation and Grounded Question Answering for Indian-Language Documents on One Machine

Ganesh F Bhovivaddar
B.E. Artificial intelligence and
Machine Learning,
Adichunchanagiri institute of
technology, Chikkamagaluru
ganeshfb2729@gmail.com

Manasa SR
B.E. Artificial intelligence and
Machine Learning,
Adichunchanagiri institute of
technology, Chikkamagaluru
manasasr85@gmail.com

Rishi A M
B.E. Artificial intelligence and
Machine Learning,
Adichunchanagiri institute of
technology, Chikkamagaluru
rishiam2005@gmail.com

TealSquare Technologies Private Limited
<https://tealsquare.in>

Abstract

A large amount of everyday paperwork in India exists only on paper or as a phone photograph, and much of it is written in a script that mainstream document tools handle poorly. This report presents an open pipeline that takes such a page from image to answer on a single machine: Tesseract recognises the text in English and nine Indian languages, a script-based detector identifies which languages are present, a language model reached through the ConscioSx API is used to repair recognition errors and to translate the text, and a retrieval step indexes the result so that questions can be answered from it with the supporting passages cited and questions that the documents do not cover are refused. Every component apart from the language model is open source and operates on the CPU of the machine that holds the documents. We outline each stage and the choices behind it, then evaluate the pipeline on synthetic pages rendered from a parallel corpus in all ten languages under six image conditions, on real photographs, and on retrieval and answering tasks built from the same corpus. The experiments examine how recognition accuracy varies by script and image quality, whether image preparation and model-based correction help, how the language string supplied to the recogniser should be chosen, whether an English-only embedding model can search Indic text, and how well the refusal threshold separates answerable questions from the rest. During the evaluation, the new test suite identified a sign error in the deskew step that was present in the first release. The code, test suite, benchmark pages, and every raw result are released under the Apache 2.0 licence.

1 Introduction

Forms, notices, receipts, hand-outs and legal papers in India are commonly produced in Hindi, Bengali, Tamil, Kannada and other languages whose scripts are non-Latin. When such a document needs to be searched, summarised or translated, the first step is to convert the page into text, and that step is where most tools fall short. Cloud services support the major Indian scripts

but require the document to leave the building, which makes them unsuitable for legal, medical and financial paperwork. Local tools generally cover one script at a time and often stop after producing the recognised text, leaving the user to clean, translate and interpret the output.

This report presents a pipeline that performs the whole job locally. A page enters as a photograph, a scan or a PDF and emerges as cleaned text, an English translation, and an index that answers questions about the document with citations. The components are intentionally unremarkable: Tesseract ([Smith, 2007](#); [Smith et al., 2009](#)) for recognition, a table of Unicode script ranges for language detection ([Unicode Consortium, 2024](#)), the Consciously cx-medium language model for correction, translation and answering, a 22 M-parameter sentence encoder ([Reimers and Gurevych, 2019](#); [Wang et al., 2020](#)) for retrieval, and SQLite for storage. What is less common is bringing these components together for nine Indian languages, measuring the contribution of each stage, and releasing those measurements along with the code.

The questions we aim to answer are practical ones. How accurate is recognition across the ten scripts, and how much does performance decline when the input is a low-resolution photo rather than a clean scan? Does the image preparation step, which denoises, straightens and binarises the page, justify its cost? Should the recogniser be told which languages to expect, and what happens when it is given the wrong ones? Can a small language model correct recognition errors in scripts it has seen comparatively little of during training, or does it make them worse? How accurate are its translations? Can an English sentence encoder retrieve useful information from Kannada or Hindi text, and if not, does translating the text before indexing address the problem? Finally, how effectively does a fixed similarity threshold distinguish questions that the documents can answer from those they cannot?

The contributions of this report are:

1. A description of the pipeline and the decisions that shaped it, including the fall-back behaviour when a stage is unavailable.
2. A benchmark of sixty rendered pages in ten languages under six image conditions, constructed from the FLORES-200 parallel corpus ([NLLB Team et al., 2022](#)) so that ground truth exists for every page, together with a translation, retrieval and answering evaluation derived from the same corpus.
3. Measurements of every stage, an account of a skew bug that the new test suite identified in our first release and the accuracy it cost, and a discussion of what the results imply for anyone building a similar system.

2 Related Work

Recognition of Indic scripts. Tesseract has supported multilingual recognition since its adaptation to non-Latin scripts ([Smith et al., 2009](#)), and version 4 replaced the classical character classifier with an LSTM line recogniser that provides trained data for all of the languages used here. [Mathew et al. \(2016\)](#) trained a single LSTM recogniser for several Indic scripts and found that Indic text is harder to recognise than Latin text, mainly because of conjunct consonants and vowel signs that can appear above, below or on both sides of a base character. [Mathew et al. \(2025\)](#) released the Mozhi dataset of 1.2 million annotated word images in thirteen Indian languages and models that outperformed Tesseract in eight of them. We selected Tesseract rather than a research recogniser because it can be installed with one command across platforms, while the results in Section 5 show the level of performance that a stronger recogniser would need to surpass.

Correcting recognition output with language models. Post-OCR correction with large language models has been investigated mostly on historical English. [Thomas et al. \(2024\)](#)

instruction-tuned Llama 2 on nineteenth-century British newspapers and reduced the character error rate by more than half, and Bourne (2024) showed that a prompted model can use surrounding context to correct errors a character model cannot. Both studies focused on text in a language that dominates model training data, and the first fine-tuned its model for the task. We examine the same question using Consciousnx cx-medium as it comes, on scripts that appear much less often in web text.

Translation and its evaluation. The FLORES-200 corpus (NLLB Team et al., 2022) provides human translations of the same sentences in more than two hundred languages, including all ten used here, which allows us to score translation against a reference and, equally usefully, render the same content in ten scripts for the recognition benchmark. IndicTrans2 (Gala et al., 2023) is the strongest open translation system for Indian languages; however, we use the general-purpose language model because it is already loaded for correction and answering, and we assess the resulting effect on translation quality. Translation quality is evaluated with chrF (Popović, 2015), which is character-based and therefore does not depend on a tokeniser for each script, computed with sacreBLEU (Post, 2018).

Retrieval-augmented answering. Retrieval-augmented generation (Lewis et al., 2020) conditions the answer on passages retrieved from a corpus. Retrieved passages that are irrelevant reduce answer quality (Cuconasu et al., 2024; Yoran et al., 2024), which is why the pipeline refuses to answer when retrieval is weak rather than passing weak passages to the model. The sentence encoder used for retrieval, all-MiniLM-L6-v2 (Sentence Transformers contributors, 2021), was trained on English pairs; language-agnostic encoders such as LaBSE (Feng et al., 2022) exist, and we discuss in Section 6 why the pipeline translates before indexing rather than switching encoders.

3 The Pipeline

Figure 1 illustrates the stages. The browser interface, the HTTP API and the command line all use the same function, so a document is processed in the same way regardless of how it arrives.

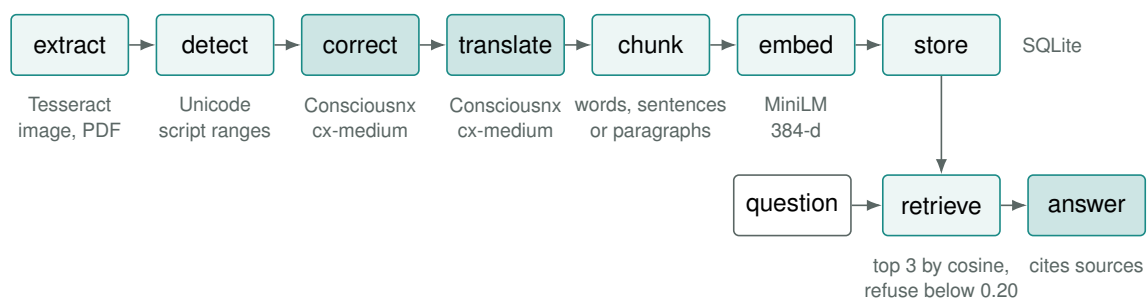


Figure 1: The stages a document passes through. Shaded stages use the language model; the remaining stages run on the CPU without any model.

3.1 Extraction

Before Tesseract processes an image, it can go through an optional preparation step that includes grayscale conversion, cubic upscaling when the page is shorter than 1000 pixels (limited to three times), non-local means denoising (Buades et al., 2005), skew estimation using the minimum-area rectangle around the ink, rotation when the estimated skew is between a quarter of a degree and fifteen degrees, and Otsu thresholding (Otsu, 1979). Tesseract 5 then recognises the page using the language string the caller supplied, or the configured default when none was provided. The

average of the per-word confidence values reported by Tesseract is retained as a rough measure of quality.

PDFs are processed page by page. A page whose text layer contains at least forty characters is treated as born-digital and its text is used as is; otherwise the page is rendered at 200 dpi and recognised like an image. A fifty-page cap prevents a stray upload from tying up the machine.

3.2 Language detection

Each character that is neither a space nor a digit is checked against a table of Unicode block ranges for the supported scripts, following the script property specified in Unicode Standard Annex 24 ([Unicode Consortium, 2024](#)). The share of each script is reported, scripts below five percent of the characters are dropped, and if nothing survives the threshold the most common script is reported anyway so that a short string still receives an answer. Devanagari is shared by Hindi, Marathi, Nepali and Sanskrit, so a Hindi detection carries a note to that effect. The same table converts the detected scripts into a Tesseract language string, enabling a two-pass approach: first recognise the page using a broad default, then detect the scripts and run recognition again using only the scripts that are present.

3.3 Correction and translation

Both stages send one prompt to the language model. The correction prompt instructs the model to correct only misrecognised characters, keep the original language, preserve line breaks, never invent content, and return only the text. The translation prompt requests fluent output in the target language with names, numbers and dates left unchanged. Either stage can fail if the model cannot be reached; in this situation, correction returns the original text with a note, while translation returns nothing, thereby preventing the source text from being presented as a translation.

3.4 Indexing and answering

Corrected text is divided into chunks by one of three strategies: fixed windows of words with an overlap, sentences packed up to a word limit (with the danda and double danda recognised as sentence ends), or paragraphs packed in the same way. Each chunk is embedded with all-MiniLM-L6-v2 into a normalised 384-dimensional vector and stored as bytes in SQLite alongside its text. The question is embedded in the same way and compared with every stored vector using a dot product, which is equivalent to cosine similarity for unit vectors; the vectors are kept in memory as a single matrix so that the comparison can be performed as one matrix-vector product. The top three chunks form the context. If the best score is below a threshold of 0.20, the pipeline states that the information is not in the documents and never calls the model. Otherwise, the model is instructed to answer only from the provided context, cite the sources it uses by number, and return a fixed sentence when the answer is not contained in the context; this sentence is then identified and recorded as a refusal.

3.5 Test suite

The test suite contains 114 unit tests. Seventy-four tests come from the first release and cover chunking, detection, the store, PDF handling, language validation and fall-back paths; the forty added for this report cover image preparation, extraction using a live Tesseract, indexing and answering with the model and encoder stubbed, and upload classification. The complete suite runs in about twelve seconds. One of the new tests failed against the first release and resulted in the finding in Section 5.2.

4 Experimental Setup

4.1 Benchmark pages

We required ground-truth data covering ten scripts, but we could not find a public page-level dataset that included all of them under a permissive licence, so we created our own pages. For each language we selected sentences from the dev split of FLORES-200 (NLLB Team et al., 2022), which contains the same 997 Wikinews and Wikipedia sentences translated by professionals into every language we support, and grouped them into six pages of about 110 words. Pages were rendered at 1654 by 2339 pixels, the size of an A4 sheet scanned at 200 dpi, in 34-pixel Nirmala UI, using the Windows GDI+ text renderer because it shapes conjuncts and vowel signs correctly in each of the nine Indic scripts (the common Python imaging libraries do not without an extra shaping engine). Each clean page was then modified in five ways to represent conditions under which documents may actually be received (Table 1), resulting in 360 test images. Figure 2 displays two of them.

Table 1: Image conditions applied to each benchmark page.

Condition	What was done to the clean rendering
Clean	Nothing
Low resolution	Scaled to 45% of its size, about 90 dpi
Noisy	Gaussian blur (3 px) and Gaussian noise ($\sigma = 25$)
Skewed	Rotated by 4 degrees
JPEG	Saved at JPEG quality 20 and decoded again
Photo	Rotated by -3 degrees, brightness gradient down to 55% across the page, blur (5 px), noise ($\sigma = 12$)

4.2 Metrics

Recognition is evaluated by character error rate (CER) and word error rate (WER), the Levenshtein distance (Levenshtein, 1966) between the recognised and the reference text divided by the reference length, computed with jiwer (jiwer contributors, 2026) after Unicode NFC normalisation, removal of zero-width joiners, and whitespace collapsing. A code point is treated as a character for this purpose, so a wrong vowel sign counts as one error even though it changes the appearance of the whole syllable. Translation is evaluated with sentence-level chrF (Popović, 2015). Retrieval is evaluated by whether the sentence a question was written from appears within any of the top k retrieved chunks. Answers are evaluated two ways: whether the gold answer string appears in the reply, and the verdict of the same Consciousnx model asked whether the reply states the same fact as the gold answer.

4.3 Models and hardware

Table 2 presents the components. The language model used is Consciousnx cx-medium, accessed through the provider’s chat completion API in the same manner as the application, using its own prompts, system messages and temperatures (0.1 for correction and answering, 0.2 for translation), with one request per call and no retries. Tesseract 5.4.0 was used with the “fast” trained data that the launcher retrieves by default.

All components except the language model operated on a single laptop with an Intel Core i7-11800H, eight cores and 16 GB of memory, running Windows 11. Recognition and embedding ran on the CPU; the model stages were sent to Consciousnx and returned. The recognition and translation benchmarks were run concurrently, so the recognition timings in Section 5 were measured while the machine was under additional load and should primarily be compared with one another.

ಅಂದಾಜು ತಲಾ ಒಂದು ಯು.ಎಸ್ ಸೆಂಟ್ ನಿಂದ ಪ್ರಮಾಣಿತ ಇಂಕ್‌ಜೆಟ್ ಪ್ರಿಂಟರುಗಳನ್ನು ಬಳಸಿ ಉತ್ಪಾದಿಸಬಹುದಾದ ಸಣ್ಣ ಪ್ರಿಂಟ್ ಮಾಡಬಹುದಾದ ಚಿಪ್ ಆಗಿರುವ ಸೆಲ್‌ಗಳ ವಿಧವನ್ನು ಆಯೋಜಿಸುವ ಹೊಸ ಪತ್ರೆ ಪರಿಕರದ ಅನ್ವೇಷಣೆಯ ಫೋಷಣೆಯನ್ನು ಸ್ಯಾನ್‌ಫೋರ್ಡ್ ವಿಶ್ವವಿದ್ಯಾಲಯದ ಜಿಷ್ಠ ಶಾಲೆಯ ವಿದ್ಯಾರ್ಥಿಗಳು ಸೋಮವಾರ ಮಾಡಿದ್ದಾರೆ. ಪ್ರಮುಖ ಸಂಶೋಧಕರು ಹೇಳುವಂತೆ, ಇದರಿಂದ ಕಡಿಮೆ ಆದಾಯ ಇರುವ ದೇಶಗಳ ರೋಗಿಗಳಲ್ಲಿ ಮತ್ತು ಶ್ರೀಮಂತ ರಾಷ್ಟ್ರಗಳಲ್ಲಿ ಇರುವುದಕ್ಕಿಂತ ಅರ್ಧದಷ್ಟು ಬದುಕುಳಿಯುವ ಪ್ರಮಾಣ ಇರುವ ಸ್ವನ ಕ್ಯಾನ್ಸರ್ ರೋಗಿಗಳಲ್ಲಿ, ಕ್ಯಾನ್ಸರ್, ಕ್ಷಯರೋಗ, HIV ಮತ್ತು ಮಲೇರಿಯಾ ರೋಗಗಳನ್ನು ಆರಂಭಿಕ ಹಂತದಲ್ಲೇ ಪತ್ತೆ ಹಚ್ಚಬಹುದು ಎಂದಿದ್ದಾರೆ. JAS 39C ಗ್ರೂಪ್‌ನ, ಸ್ಥಳೀಯ ಸಮಯ ಬೆಳಿಗ್ಗೆ 9:30 ಕ್ಕೆ (0230 UTC) ರನ್ ವೇ ಮೇಲೆ ಅಪ್ಪಳಿಸಿ ನಂತರ ಸ್ಕೋಟಗೊಂಡಿದ್ದರಿಂದ, ವಿಮಾನ ನಿಲ್ದಾಣವನ್ನು ಕಮರ್ಷಿಯಲ್ ವಿಮಾನಗಳಿಗೆ ಮುಚ್ಚಲಾಯಿತು. ಆ ಪೈಲಟ್‌ನನ್ನು ಸ್ಯಾಡ್ರನ್ ಲೀಡರ್ ದಿಲೋಕ್ರಿತ್ ಪಟ್ಟವೀ ಎಂದು ಗುರುತಿಸಲಾಗಿದೆ. ವಿಮಾನ ನಿಲ್ದಾಣದ ಅಗ್ನಿ ಶಾಮಕ ದಳದ ಒಂದು ವ್ಯಾನು, ಸ್ಕೋಟದ ಜಾಗಕ್ಕೆ ಹೋಗುವಾಗ ಉರುಳಿತು ಎಂದು ಸ್ಥಳೀಯ ಮಾಧ್ಯಮಗಳು ವರದಿ ಮಾಡಿವೆ. 28 ವರ್ಷದ ವಿಡಾಲ್ ಮೂರು ಸೀಸನ್ ಗಳ ಮುಂಚೆ, ಸೆವಿಲ್ಲಾ ದಿಂದ ಬರ್ಕಾಗೆ ಸೇರಿಕೊಂಡಿದ್ದಾರೆ.

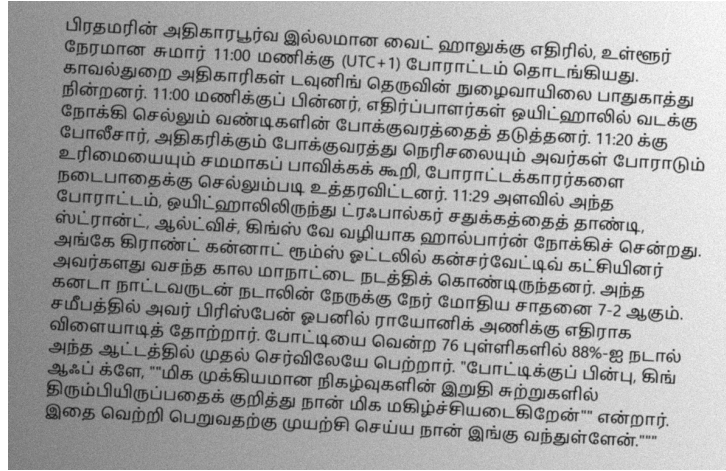


Figure 2: Two benchmark pages, cropped to the text: a clean Kannada rendering (left) and a Tamil page under the photographic condition (right).

Table 2: Components being tested.

Role	Component	Size	Licence
Recognition	Tesseract 5.4.0, tessdata_fast	10 language packs	Apache 2.0
Correction, translation, answers	Consciousnx cx-medium	hosted	Commercial
Embedding	all-MiniLM-L6-v2	22.7 M parameters	Apache 2.0
Storage	SQLite	—	Public domain

5 Results

5.1 Recognition across scripts and image conditions

Figure 3 reports the character error rate for every language and condition with image preparation on, after the deskew fix described below. On clean pages English is recognised almost perfectly and the nine Indic scripts average 5.2% CER, with Hindi, Bengali, Punjabi and Gujarati ranging between 1.3 and 1.4%, Kannada at 3.7%, Odia and Telugu around 5%, Malayalam at 8.8% and Tamil at 18.8%. The higher Tamil error rate is not caused by an issue with the glyphs. Four of the six Tamil pages score between 1 and 4%; on the other two, Tesseract’s automatic page segmentation divides the page into several blocks and emits the lines in the wrong order, which the edit distance penalises heavily even though the recognised words are mostly correct. Forcing a single-block layout brings those pages into line with the others (Section 6).

Word error rates run three to five times higher than character error rates for the agglutinative languages: on clean pages Malayalam has 8.8% CER but 39.0% WER, Kannada 3.7% against

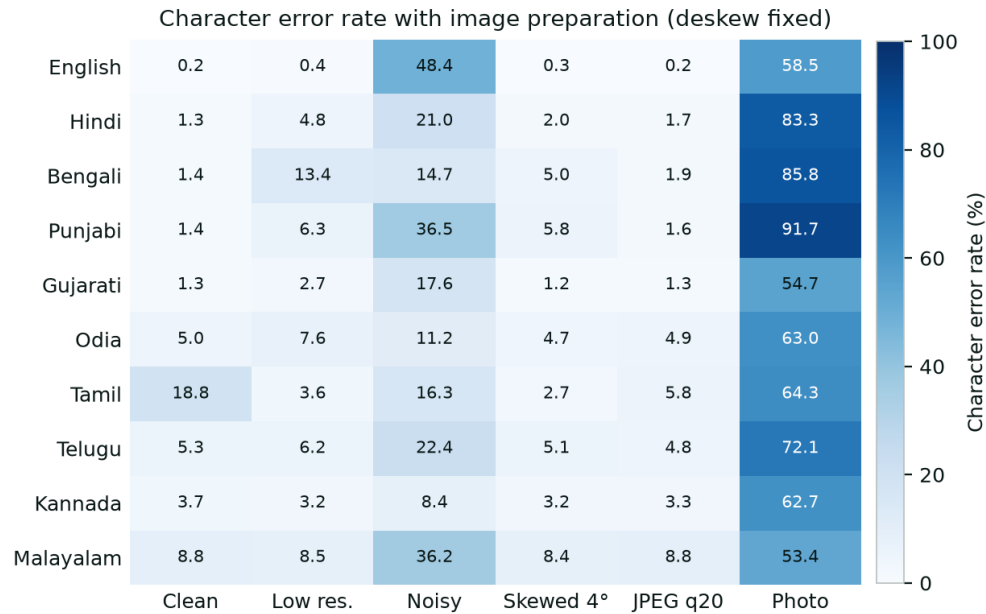


Figure 3: Character error rate (%) by language and image condition, six pages per cell, image preparation on with the deskew fix applied.

22.4%, and Tamil 18.8% against 36.6%. Words in these languages are long, so one wrong vowel sign spoils a word that would be five or six words in English. Anyone scoring an Indic recogniser by word accuracy should expect this and should not read it as a sign that the engine is failing.

Reducing the resolution to about 90 dpi costs surprisingly little for most scripts (Bengali is the exception, rising from 1.4 to 13.4%), and JPEG compression at quality 20 has almost no effect. The two conditions that cause problems are noise and the photo condition, and the next subsection separates the different reasons.

5.2 What image preparation does, and the deskew bug

Figure 4 compares three settings using the same 360 images: no preparation, the preparation used in the first release, and the preparation following a one-line fix. The first release performed worse than doing nothing on every condition, and catastrophically so on skewed pages, where it turned a 22.2% error rate into 82.5%. The cause is a sign mistake. The skew estimator returns the page's own tilt, and the rotation matrix built from that value rotates the page by the same angle in the same direction, so a 4-degree tilt becomes an 8-degree tilt. The new deskew unit test, which rotates a synthetic page by a known angle and checks that the corrected page is level, failed on the initial run and pointed directly at the line. With the sign corrected, skewed pages fall to 3.8% CER, below the no-preparation figure, and the overall mean drops from 30.7% to 18.3%.

The fix leaves two problems unchanged. On noisy pages, preparation still increases the error rate from 3.7% to 23.3%, and on the photo condition it only reduces the error rate from 75.5% to 68.9%. Figure 5 breaks the pipeline apart on thirty pages to see which step is responsible. Grayscale conversion and upscaling make no difference. Denoising on its own is benign. Deskew with Otsu binarisation and no denoising performs well on noisy pages (5.6%) and fixes skew (4.1%). The damage on noisy pages appears only when denoising and deskew are combined: the raw noise covers the page uniformly, so the minimum-area rectangle around the ink is the page itself and the estimated angle is close to zero, but after denoising a few specks remain in the margins, the rectangle tilts to include them, and a rotation of several degrees is applied to a page that was already level. On one English page the estimate reached 4.5 degrees and recognition returned

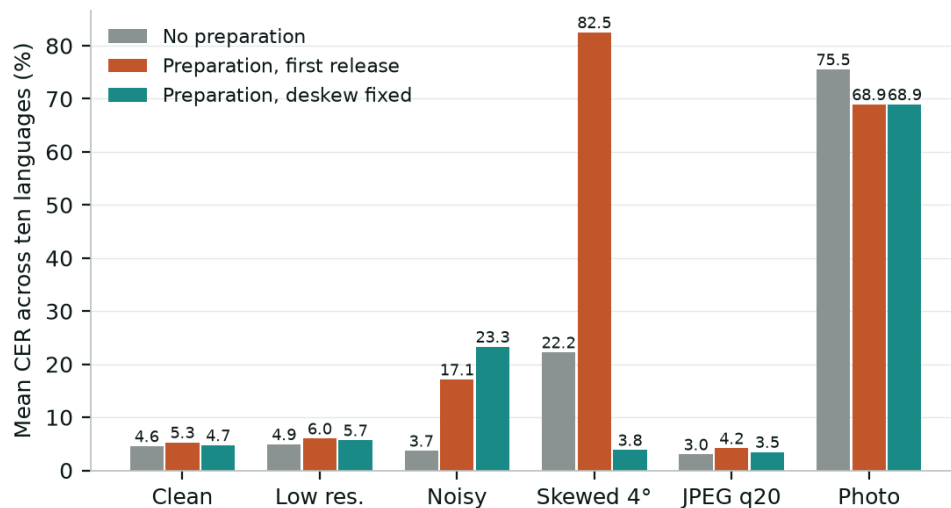


Figure 4: Mean CER across ten languages for each image condition without preparation, with the preparation of the first release, and with the deskew sign corrected.

nothing. The estimator needs to operate on text lines rather than on every ink pixel; until it does, denoising and deskew should not be chained.

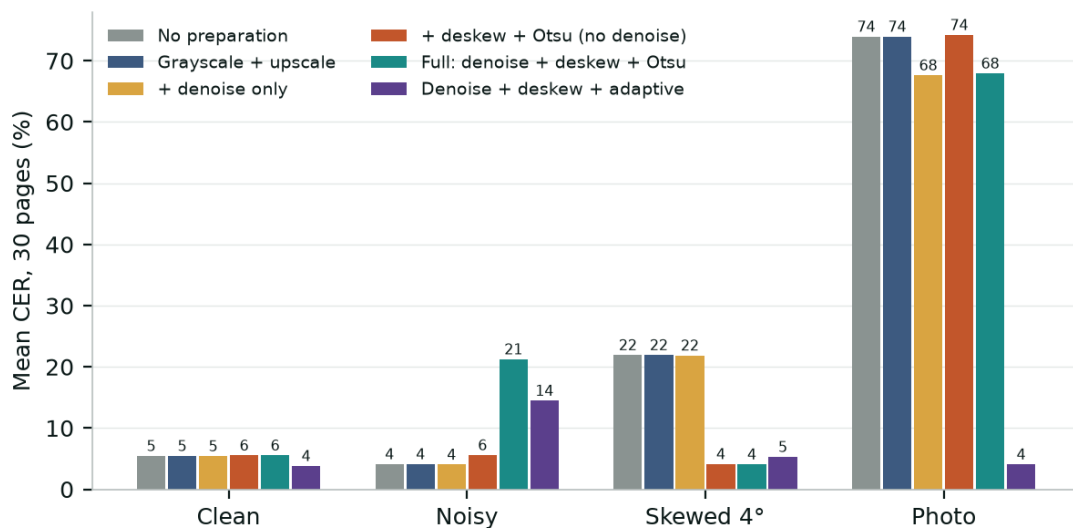


Figure 5: Preparation steps evaluated separately on thirty pages (three per language). Each bar represents the mean CER for one combination of steps; the last bar in each group replaces Otsu's global threshold with an adaptive Gaussian threshold.

The photo condition fails for a different reason. Its brightness gradient extends from full white to 55% across the page, and Otsu's method selects one threshold for the whole image, so the darker side is binarised to solid black and every character there is lost. Replacing the global threshold with an adaptive one computed in a 51-pixel window reduces the photo condition from 68% to 4.1% CER, with English at 0.3% and every Indic script below 10%. The adaptive threshold costs nothing on clean pages (3.8% against 5.4%) and handles skew as well as Otsu does, but performs worse on the synthetic noise (14.5% against 4%), because a local threshold amplifies speckle that a global one ignores. A production setting should choose the threshold from the page, for instance by measuring the background variation first.

5.3 Is Tesseract's confidence a useful quality signal?

The pipeline stores the mean per-word confidence Tesseract reports, and the interface displays it as an indication of how much to trust the text. Figure 6 compares this value with the measured CER for the 354 pages that provided a confidence score. The correlation is -0.33 : pages with confidence above 90% are almost always good, but 47 pages show confidence above 80% while losing more than a fifth of their characters. All of these come from the photo and noisy conditions, where the binarisation removed whole regions of the page. Tesseract is confident about the words it found and has no way to know about the words it never saw. The confidence is worth displaying, but a low value is a much stronger signal than a high one.



Figure 6: Mean word confidence provided by Tesseract against the measured character error rate, one point per page. Six pages for which Tesseract reported no confidence are omitted.

5.4 Language detection

Detection is precise on clean text. Across all 9,970 FLORES sentences the dominant script was identified correctly in every case but one (a Punjabi sentence consisting almost entirely of Latin letters), and the secondary detections were all genuine: Telugu sentences, for instance, contain Latin abbreviations in 77 of 997 cases. Figure 7 presents two-script documents mixed at known character ratios. The reported share follows the true share to within a percentage point across the whole range, and the 5% reporting threshold behaves as designed: a second script at 2% is dropped, at 5% it is reported in seven of ten trials, and at 10% or more it is always reported.

The detector is similarly reliable on recognised text, although there is one qualification. Across the 360 pages with preparation on, the top detected language was correct on every page that produced text, and wrong only on the thirteen pages (all noisy or photo) where recognition returned nothing at all. Noise in the recognised characters does not confuse a script-range lookup, because a misrecognised Kannada letter remains, almost always, a Kannada letter.

5.5 Choosing the recogniser's language string

Tesseract needs to know which languages to load, so the detector module includes a helper for a two-pass approach: first perform recognition with the default string, detect the scripts in the output, and then run recognition again using only those scripts. Table 3 evaluates that advice on thirty clean pages. Specifying the correct language alone gives 5.5% CER in two seconds a page. The default string (English, Kannada, Tamil, Telugu) works for those four languages and is

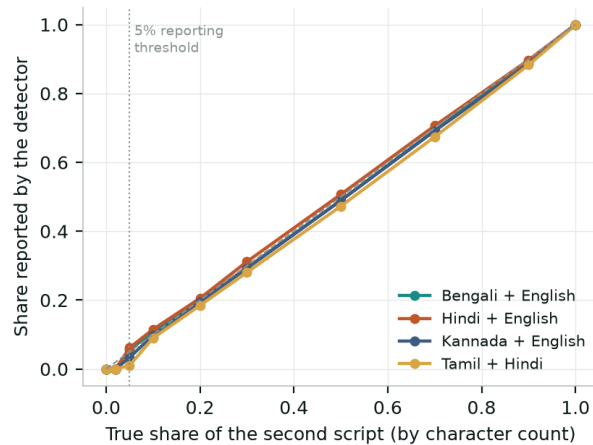


Figure 7: Two-script documents at known mixing ratios, five trials per point. The dashed line is perfect agreement.

useless for the other six, where CER is around 80% because the engine renders Hindi or Bengali as the nearest Tamil and Kannada shapes. Detecting from that output and re-running does not help, since the garbage is itself in Tamil and Kannada script and the detector suggests exactly those languages. Loading all ten languages at once costs 8.3% CER and takes four times as long, because the engine now has ten alphabets to confuse; Odia suffers most (15.4%), being visually close to Bengali. The approach that works is to run the first pass with all ten languages, detect, and re-run with the narrowed string. The detector selected the correct single language on 29 of the 30 pages (on one Telugu page it kept English as well, which cost nothing), and the second pass matched the oracle exactly at 5.5% CER. The trade-off is time: 10.9 seconds a page against 2.0, because the first pass alone takes eight. For a document store where each page is processed once and searched many times, that is a fair trade; for the interactive recognition endpoint, which stores nothing, the user should be prompted for the language.

Table 3: Methods of choosing the language string, thirty clean pages (three per language).

Language string	Mean CER (%)	Seconds per page
Correct language only (oracle)	5.5	2.0
Default string (eng+kan+tam+tel)	53.8	4.6
Default, detect, re-run with the suggestion	53.3	8.0
All ten languages	8.3	8.3
All ten, detect, re-run with the suggestion	5.5	10.9

5.6 Does model-based correction improve results?

Figure 9 addresses the question the correction stage was designed to answer. The recognised text of 120 pages (four per language, clean, noisy and low-resolution) was processed through the correction prompt and scored again. On average it made little difference: clean pages went from 4.9% to 5.1% CER, noisy pages from 24.7% to 24.9%, and low-resolution pages from 5.7% to 6.1%. Thirty-eight pages improved, by a median of 0.5 points; 69 became worse, by a median of 0.6 points; thirteen remained unchanged; and four noisy pages had produced no text for the model to correct. Only Kannada (6.3% to 5.5%) and Malayalam (22.5% to 21.5%) showed an average gain. The step takes ten seconds per page, with nearly all of that time spent waiting for the model.

Examining the output helps explain the numbers. On clean pages the model mostly returns the input unchanged, which is appropriate and costs nothing but time. When it does make changes,

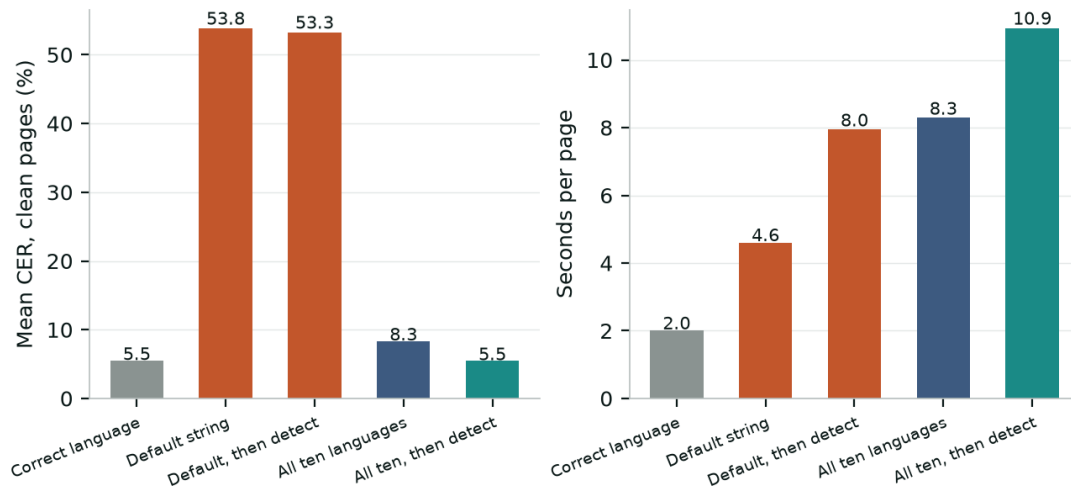


Figure 8: Accuracy and cost of five Methods for choosing the language string, thirty clean pages.

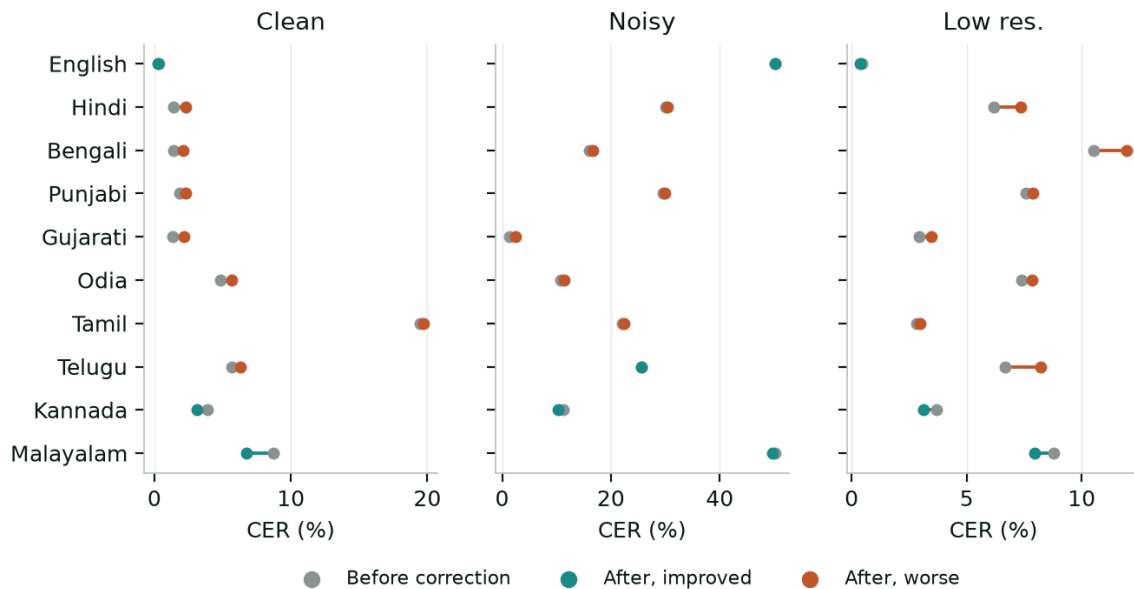


Figure 9: Character error rate before (grey) and after model-based correction, by language, across three image conditions. Teal marks a language whose mean improved, rust one that became worse.

it tends to normalise the text rather than actually repair it: it replaces Kannada digits with ASCII ones, removes a stray punctuation mark, or rewrites a word it does not recognise as a more common one, and each of those is an edit away from the reference. On noisy English pages it shortens the text by a sixth, removing fragments it judges unreadable, which the prompt permits but the metric counts as deletions. What it does not do, for any script, is reconstruct the missing or wrong vowel signs that account for most of the genuine errors. The model does not know enough Bengali or Odia orthography to distinguish a plausible misspelling from a rare word, and it was never shown what Tesseract’s errors look like. The correction studies that report large gains (Bourne, 2024; Thomas et al., 2024) fine-tuned much larger models on paired noisy and clean text; the zero-shot version of that approach, with this model and in these scripts, does not work. The stage remains in the pipeline because it can be switched per request, and we recommend that the default be turned off.

5.7 Translation Results

Table 4 and Figure 10 assess the Consciousnx model’s translations using forty FLORES sentences per language. For translation into English the model is consistent across all nine languages, with corpus chrF between 51.7 (Tamil, Telugu) and 58.7 (Hindi) and BLEU between 21 and 29. Every number, date and identifier in the reference remained unchanged in the translation for all ninety sentences that contained one, so the prompt’s instruction to leave them alone is being followed. The typical failure is a plausible sentence with the incorrect sense of one word: a Kannada sentence about seeing pebbles on a lake bottom became “you can tirelessly see the rocks or coral below”, which scores 24 chrF against a reference it partly matches. For comparison, dedicated translation systems report chrF in the sixties on this corpus for these languages (Gala et al., 2023; NLLB Team et al., 2022); the general-purpose model sacrifices roughly ten points for the convenience of a single model that also corrects and answers.

Table 4: Translation quality of Consciousnx cx-medium on FLORES-200 dev sentences. For English output: 40 sentences per language. From English: 20 sentences. BLEU uses sacreBLEU’s default tokeniser for English output and no tokeniser for Indic output.

Direction	Language	chrF (corpus)	chrF (sentence mean)	BLEU	s/sentence
Into English	Hindi	58.7	57.5	29.3	2.8
	Bengali	53.5	53.1	22.5	2.8
	Punjabi	53.3	53.4	25.4	2.8
	Gujarati	56.4	55.4	27.2	2.8
	Odia	54.6	54.4	25.6	2.8
	Tamil	51.7	51.1	21.6	2.7
	Telugu	51.7	50.8	21.1	2.8
	Kannada	53.7	54.6	22.8	2.8
	Malayalam	52.7	53.4	21.5	2.8
From English	Hindi	44.4	43.3	14.7	3.1
	Kannada	36.2	37.7	3.9	3.5

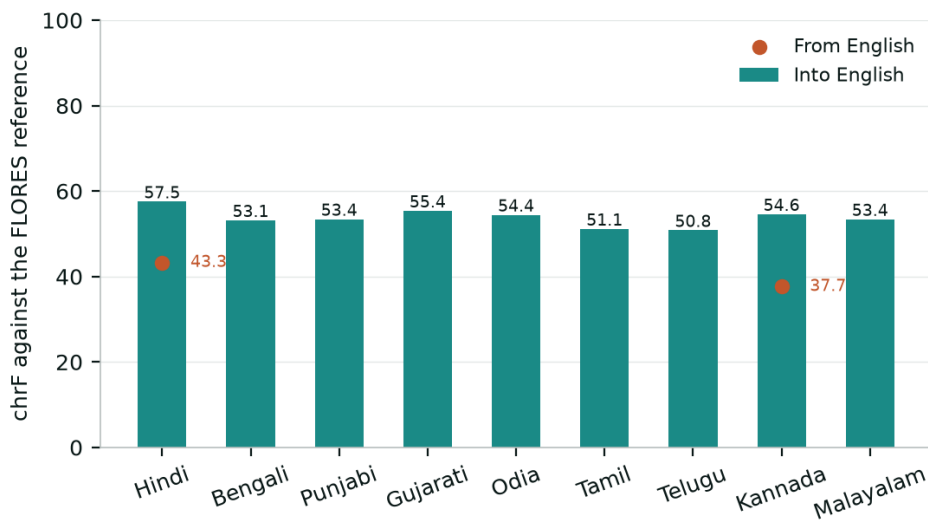


Figure 10: Sentence-level chrF by language. Bars: translations into English. Points: translations from English into Hindi and Kannada.

The reverse direction performs much worse. English into Hindi reaches 44 chrF, while English into Kannada reaches only 36 chrF with a BLEU of 3.9. The Kannada output is grammatical but relies on transliterated English (“official” rendered phonetically rather than as the Kannada

word) and on constructions a native writer would not choose. The application lets users select any supported language as the translation target, and this result suggests that feature should be labelled as a draft for the Dravidian languages until a dedicated translation model is wired in.

5.8 Retrieval, refusal, and answering

Chunking. Figure 11 illustrates, for sixty model-written questions over a 240-sentence English corpus, how often the sentence a question was written from appears in the top k chunks. Our default, 250-word windows with a 50-word overlap, retrieves the source in the top three 72% of the time. Smaller sentence-aligned chunks perform much better: packing sentences up to 40 words puts the source in the top three 93% of the time and in the top one 78% of the time. Fixed 40-word windows, by contrast, are the weakest setting (57% at top three), because they cut sentences in half and the embedding of a half sentence matches nothing well. Chunk boundaries have more impact than chunk size. The recommended default from this experiment is the sentence strategy at 40 to 80 words.

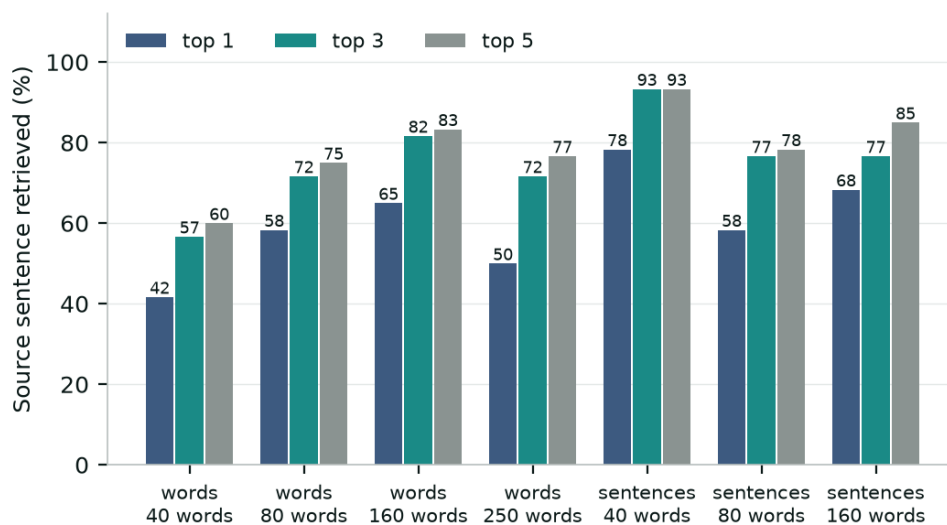


Figure 11: Retrieval of the source sentence for sixty questions across seven chunking settings. The default uses 250-word windows.

Searching Indic text. The embedding model was trained on English text. Figure 12 evaluates the same sixty English questions against the Kannada and Hindi versions of the same corpus. When indexed in the original script, retrieval barely beats chance: the source sentence is in the top three 13% of the time for Kannada and 7% for Hindi, and the tokeniser turns each Indic word into a long sequence of byte fragments that carry no meaning the encoder was trained on. When indexed after machine translation by the Consciousnx model, the same corpus reaches 78% for Kannada and 67% for Hindi, compared with 77% for the human English reference under the same chunker. The translation step, in other words, restores essentially all of the retrieval quality for Kannada and most of it for Hindi. This provides the strongest argument in the report for translating documents at indexing time rather than storing them in their original script, at least while the encoder remains English-only.

The refusal threshold. Figure 13 plots the highest retrieval score for three groups of questions: the sixty answerable ones, thirty written from Wikinews sentences that are not in the corpus, and twenty general-knowledge questions. Answerable questions score between 0.19 and 0.78 (median

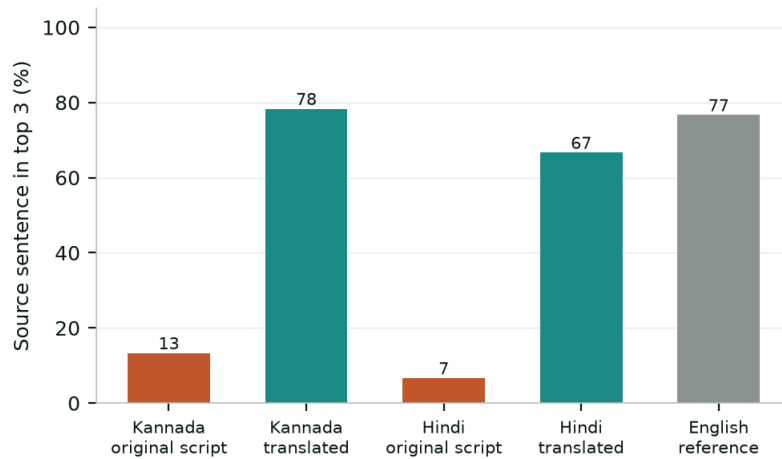


Figure 12: Retrieval of the source sentence in the top three when the corpus is indexed in Kannada or Hindi script, following machine translation into English, or as the human English reference. Sentence chunks of 80 words throughout.

0.48); the two unanswerable groups range between 0.12 and 0.39 (medians 0.24 and 0.23). The default threshold of 0.20 refuses one answerable question and allows 36 of the 50 unanswerable ones through to the model. A threshold of 0.30 would refuse six answerable questions and 37 of the 50 unanswerable ones; at 0.40 no unanswerable question passes but 19 answerable questions are lost. There is no clear separation, because a question about a news story that is merely similar to one in the corpus looks, to a sentence encoder, a lot like a question the corpus answers.

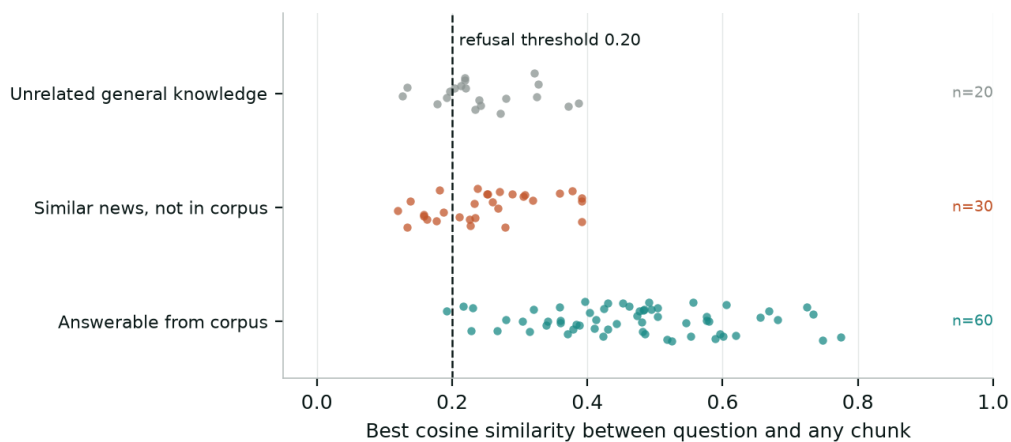


Figure 13: Highest cosine similarity between each question and any indexed chunk, for answerable questions and two kinds of unanswerable ones. The dashed line represents the default refusal threshold.

Answering. The second line of defence is what makes the threshold more useful. Table 5 takes all 110 questions through the full answer path. All fifty of the unanswerable questions were refused: the threshold caught fourteen, and the model, instructed to reply with a fixed sentence when the context does not contain the answer, caught the other 36 without inventing anything. Of the sixty answerable questions, 44 were judged correct, 5 were answered incorrectly, and 11 were refused. Six of the refusals are questions the generator wrote poorly (“What is it?”, “What are they doing?”), which no retriever can serve; the other five had the correct passage in the context and the model declined anyway. The five incorrect answers are all cases where the top chunk contained a different fact of the same kind, for example a different goal count from the same

match report. Answers cited a source in all 49 of the 49 grounded replies.

Table 5: Outcome of the full answering path for 110 questions. A reply is counted as correct when the Consciousnx model acting as judge accepts it or when the gold string appears in it.

Question group	N	Refused by		Correct	Wrong	s/question
		threshold	model			
Answerable from corpus	60	1	10	44	5	2.9
Similar news, not in corpus	30	9	21	0	0	2.0
Unrelated general knowledge	20	5	15	0	0	2.1

5.9 Two real photographs

The synthetic pages contain clean text rendered in a single font. Figure 14 presents two photographs of the kind users actually upload: a page from a decades-old Kannada book, yellowed, with the type worn and a stamp across the middle, and a page of handwritten Kannada and English lecture notes. There is no reference text for either, so Table 6 records what each stage produced.

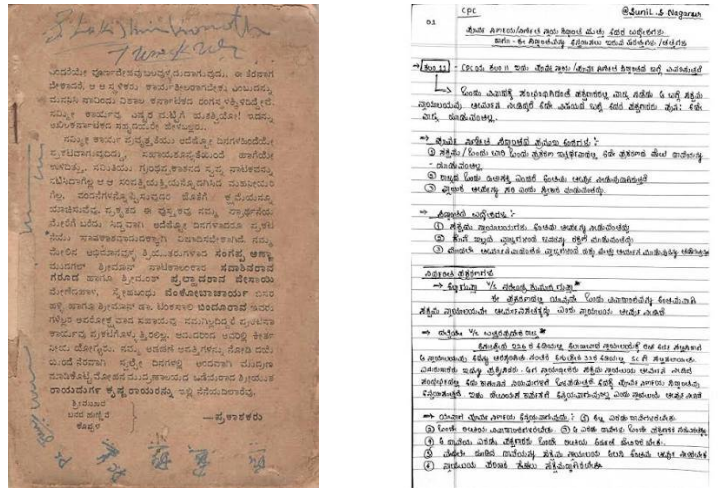


Figure 14: The two photographs used in the case study: an old printed Kannada page (left) and handwritten Kannada and English lecture notes (right).

Table 6: Pipeline results on two real photographs, with and without image preparation.

Photograph	Preparation	Confidence	Characters	Detected (share)	Seconds
Printed page	on	0.45	1014	Kannada 0.86, English 0.14	24
Printed page	off	0.38	918	Kannada 0.80, English 0.20	25
Handwritten	on	0.35	1181	Kannada 0.57, English 0.43	30
Handwritten	off	0.34	575	English 0.62, Kannada 0.38	18

Neither page could be read. On the printed page Tesseract returned about a thousand characters, of which perhaps a third are real words; on the handwritten page almost nothing was. The confidence values, 0.35 to 0.45, are well below anything observed on the synthetic pages and would have been the appropriate cue to stop, which supports the reading of Section 5.3: low confidence is informative. Detection still performed its job, reporting Kannada as the main script for the printed page, and the mix of Kannada and Latin fragments on the handwritten page is an accurate description of what the recogniser produced.

The stages after recognition behaved as instructed by their prompts, which revealed another problem. Correction returned the text largely unchanged. Translation, given a page of fragments,

produced fluent English about a full moon wet with dew and memories belonging to a country; none of it appears in the source. In one of the four runs, the model echoed the Kannada input instead, which the pipeline would have shown as a translation. A translation stage that always produces something is a risk when the input is garbage, and the fix belongs upstream: when confidence is this low, the interface should say the page could not be read and skip the model stages that would otherwise turn the noise into prose.

5.10 Where the time goes

Table 7 collects per-stage timings from the runs above, and Figure 15 presents them on a log scale. The components outside the language model are fast: CPU recognition takes one to two seconds per page, while detection and search require very little time and embedding a chunk takes about twenty milliseconds. The model stages account for the cost. Correction takes ten seconds per page, and the two-pass language selection of Section 5.5 adds nine seconds of recognition. A page that is recognised, detected, translated sentence by sentence and indexed therefore takes roughly half a minute on this laptop, most of it in translation; a question against an indexed corpus is answered in about three seconds, or in a tenth of a second when the threshold refuses it before the model is called.

Table 7: Time per stage, measured on the benchmark laptop. Model stages include the round trip to Consciousnx; the remaining stages ran on the CPU.

Stage	Seconds
Recognition of one A4 page at 200 dpi, no preparation	1.15
Recognition with preparation	1.78
Script detection, one page	0.001
Chunking, 240 sentences	0.001
Embedding, one 80-word chunk	0.022
Embedding, one question	0.009
Indexing 303 chunks (997 sentences), end to end	4.0
Search over 303 chunks	0.0001
Translation, one sentence (Consciousnx)	2.8
Correction, one page (Consciousnx)	10.0
Answer, refused at the threshold	0.11
Answer, with the Consciousnx model	3.0

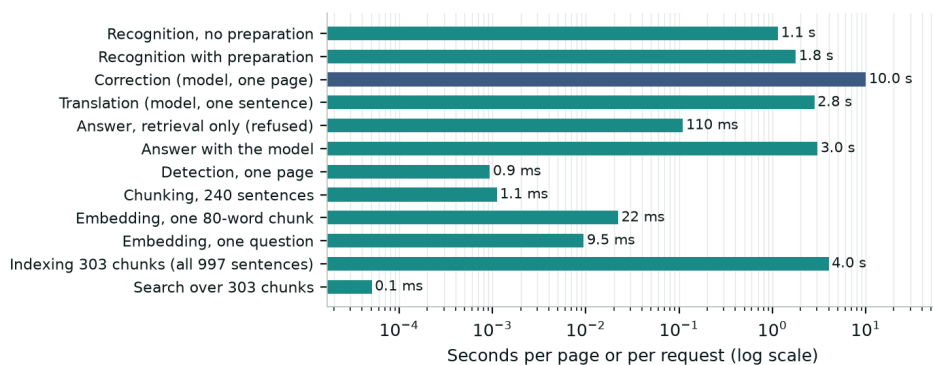


Figure 15: Seconds per stage on a log scale. Teal marks stages below five seconds.

6 Discussion

What the recogniser needs from the code around it. The largest effects in this report have nothing to do with the recogniser’s own accuracy. A wrong sign in one line quadrupled the error rate on skewed pages; a global threshold lost half of every photographed page; a default language string made six languages unreadable; a page-segmentation mode turned two clean Tamil pages into 70% error. Each of these was invisible from inside the application, because the recognised text was still text and Tesseract still reported a confidence of 80% or better. The benchmark pages made them visible in an afternoon. The practical lesson is that a pipeline built from mature components still needs ground truth of its own, and that rendering it from a parallel corpus is cheap enough that there is no excuse not to.

Page segmentation. The two Tamil pages that scored 70% and 34% CER were re-run with Tesseract’s single-block segmentation mode rather than the automatic one: they fell to 2.8% and 6.9%, and a third Tamil page went from 4.1% to 1.6%, while Malayalam and Odia pages remained unchanged. The automatic mode searches for columns and tables, but on a page of dense Tamil prose it can sometimes identify columns that are not actually present. For single-column documents, which make up most of what this pipeline sees, the single-block mode is the safer default, and it requires only a one-word change to the Tesseract configuration.

Recommended settings after these experiments. The measurements point to a small set of changes. The first is already in the code; the remaining recommendations for the next release are: deskew with the corrected sign; estimate skew from the denoised image but rotate only when the estimate comes from a page whose ink forms a block, or skip the deskew step when denoising is enabled; use an adaptive threshold when the page background is uneven and Otsu when it is flat; run the recogniser with single-block segmentation for single-column input; when the caller does not provide a language, run the first pass with all ten and narrow it using the detector; chunk by sentences at 40 to 80 words; translate before indexing; disable model-based correction by default; and refuse to run the model stages when recognition confidence is below about 0.5.

The refusal threshold and the model. No cosine-similarity threshold separates “the corpus answers this” from “the corpus has a story like this”. The pipeline handles this by instructing the model to refuse when the context does not contain the answer, and with this model that instruction worked on all fifty unanswerable questions. That is a stronger result than we expected from a general-purpose model, and this is why the threshold can remain low enough not to lose answerable questions. This result should also be rechecked whenever the model is changed, because a more eager model could produce unsupported answers.

Translation as the bridge to search. An English-only encoder cannot search Indic text, and the solution that requires nothing new is already in the pipeline: translate at indexing time and search the translation. The difference between the translated Kannada corpus and the human English reference was just one percentage point. A language-agnostic encoder such as LaBSE (Feng et al., 2022) would allow searching in the original script and asking in the original language, which the current design cannot do, but it would replace a 22 M-parameter model with a 470 M-parameter one and would still leave the answer stage operating in whatever language the model answers best. Translating first keeps the store small and the answer stage in English, but the chrF scores show that the translation is suitable for search rather than serving as a replacement for a dedicated translation model when translation itself is the product.

Why correction did not help. The correction stage was designed on the assumption that a language model can read Indic scripts well enough to see where a vowel sign is missing. The results show that this assumption does not hold for this model when it is used without examples. Correction studies that succeed fine-tune on paired noisy and clean text, or provide the model with examples of the recogniser’s own mistakes. For this pipeline, improving the input is therefore a less expensive way to reduce errors, since the ablation shows gains of ten to seventy points on the important conditions, compared with only a fraction of a point from correction.

7 Limitations

1. The benchmark pages are generated in one font at one size. Real documents use many typefaces, at many sizes, with tables and headings, and the error rates here are a lower bound on what those may show. The two photographs in Section 5 provide a sense of how large the gap can be.
2. The synthetic degradations are substitutes. Real noise, real skew and real lighting do not come from a Gaussian, and the photo condition in particular is a single, fairly mild example of a large family.
3. Tesseract was run with the “fast” trained data because that represents what the launcher provides. The “best” data is larger and slower, and could alter the clean-page numbers, especially for Tamil and Malayalam.
4. The model stages depend on a hosted service, so their timings include the network round trip and may vary with the service load. The accuracy results are also tied to the cx-medium model as served in September 2026, and a later model version could alter them in either direction.
5. Retrieval and answering were evaluated on English text using questions generated by the same model that produced the answers. Human-written questions may match the wording of the source less closely and could therefore reduce the hit rates.
6. Translation was evaluated with automatic metrics on forty sentences per language. chrF rewards overlap with one reference and does not capture fluency errors that a reader would notice.
7. The photo case study contains two images and no ground truth.
8. The refusal experiment used a corpus of 240 sentences. Larger corpora are likely to contain more near-matches, which could increase the overlap between the score distributions shown in Figure 13.

8 Conclusion

An ordinary laptop with open-source components and one hosted model API can take a photographed or scanned page in any of nine Indian languages and turn it into searchable, translated text with cited answers, and this report measures how well each part of that pipeline works. Tesseract reads clean pages in all ten scripts with a character error rate under 6% for eight of them, and the two exceptions are a segmentation setting and a weaker trained-data pack rather than the scripts themselves. Image preparation is the stage that decides whether a photographed page is readable at all: with a sign bug fixed and an adaptive threshold in place it takes the photo condition from roughly 70% error to 4%, while the first release made every condition worse. The script detector is exact and cheap, but it can only choose a language string well if the first recognition pass was run with every language loaded. Consciousnx cx-medium translates into English at a usable if not professional level, refuses unanswerable questions reliably, answers most answerable ones, and does not repair recognition errors. An English sentence encoder cannot search Indic text, and translating before indexing closes that gap almost entirely.

The changes indicated by these results are small: a sign, a threshold, a segmentation mode, a language string, a chunk size, and a default that is switched off. What made these issues possible to detect was a benchmark with ground truth for every script, built from a parallel corpus and a

text renderer. The next steps are a page-level test set of real documents with human transcriptions, the “best” trained data and a modern Indic recogniser evaluated through the same benchmark, a language-agnostic encoder so that questions can be asked in the document’s own language, and a small fine-tuned correction model trained on Tesseract’s actual errors in these scripts.

References

- Jonathan Bourne. CLOCR-C: Context leveraging OCR correction with pre-trained language models. *arXiv preprint arXiv:2408.17428*, 2024. URL <https://arxiv.org/abs/2408.17428>.
- Antoni Buades, Bartomeu Coll, and Jean-Michel Morel. A non-local algorithm for image denoising. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 2, pages 60–65, 2005. doi: 10.1109/CVPR.2005.38.
- Florin Cuconasu, Giovanni Trappolini, Federico Siciliano, Simone Filice, Cesare Campagnano, Yoelle Maarek, Nicola Tonellotto, and Fabrizio Silvestri. The power of noise: Redefining retrieval for RAG systems. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 719–729. ACM, 2024. doi: 10.1145/3626772.3657834.
- Fangxiaoyu Feng, Yinfei Yang, Daniel Cer, Naveen Arivazhagan, and Wei Wang. Language-agnostic BERT sentence embedding. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 878–891, 2022. doi: 10.18653/v1/2022.acl-long.62. URL <https://aclanthology.org/2022.acl-long.62/>.
- Jay Gala, Pranjal A. Chitale, Raghavan AK, Varun Gumma, Sumanth Doddapaneni, Aswanth Kumar, Janki Nawale, Anupama Sujatha, Ratish Puduppully, Vivek Raghavan, Pratyush Kumar, Mitesh M. Khapra, Raj Dabre, and Anoop Kunchukuttan. IndicTrans2: Towards high-quality and accessible machine translation models for all 22 scheduled Indian languages. *Transactions on Machine Learning Research*, 2023. URL <https://openreview.net/forum?id=vfT4YuzAYA>. arXiv:2305.16307.
- jiwer contributors. jiwer: Evaluate your speech-to-text system with similarity measures. Software, 2026. URL <https://github.com/jitsi/jiwer>. Apache 2.0. Accessed September 2026.
- Vladimir I. Levenshtein. Binary codes capable of correcting deletions, insertions, and reversals. *Soviet Physics Doklady*, 10(8):707–710, 1966.
- Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. Retrieval-augmented generation for knowledge-intensive NLP tasks. In *Advances in Neural Information Processing Systems*, volume 33, pages 9459–9474, 2020.
- Minesh Mathew, Ajeet Kumar Singh, and C. V. Jawahar. Multilingual OCR for Indic scripts. In *Proceedings of the 12th IAPR Workshop on Document Analysis Systems (DAS)*, pages 186–191. IEEE, 2016. doi: 10.1109/DAS.2016.68.
- Minesh Mathew, Ajoy Mondal, and C. V. Jawahar. Towards deployable OCR models for Indic languages. In *Pattern Recognition: 27th International Conference (ICPR 2024)*, volume 15319 of *Lecture Notes in Computer Science*. Springer, 2025. doi: 10.1007/978-3-031-78495-8_11. arXiv:2205.06740.

- NLLB Team, Marta R. Costa-jussà, James Cross, Onur Çelebi, Maha Elbayad, Kenneth Heafield, Kevin Heffernan, Elahe Kalbassi, Janice Lam, Daniel Licht, Jean Maillard, Anna Sun, Skyler Wang, Guillaume Wenzek, Al Youngblood, Bapi Akula, Loic Barrault, Gabriel Mejia Gonzalez, Prangthip Hansanti, John Hoffman, Semarley Jarrett, Kaushik Ram Sadagopan, Dirk Rowe, Shannon Spruit, Chau Tran, Pierre Andrews, Necip Fazil Ayan, Shruti Bhosale, Sergey Edunov, Angela Fan, Cynthia Gao, Vedanuj Goswami, Francisco Guzmán, Philipp Koehn, Alexandre Mourachko, Christophe Ropers, Safiyyah Saleem, Holger Schwenk, and Jeff Wang. No language left behind: Scaling human-centered machine translation. *arXiv preprint arXiv:2207.04672*, 2022. URL <https://arxiv.org/abs/2207.04672>.
- Nobuyuki Otsu. A threshold selection method from gray-level histograms. *IEEE Transactions on Systems, Man, and Cybernetics*, 9(1):62–66, 1979. doi: 10.1109/TSMC.1979.4310076.
- Maja Popović. chrF: Character n-gram F-score for automatic MT evaluation. In *Proceedings of the Tenth Workshop on Statistical Machine Translation*, pages 392–395, 2015. doi: 10.18653/v1/W15-3049. URL <https://aclanthology.org/W15-3049/>.
- Matt Post. A call for clarity in reporting BLEU scores. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pages 186–191, 2018. doi: 10.18653/v1/W18-6319. URL <https://aclanthology.org/W18-6319/>.
- Nils Reimers and Iryna Gurevych. Sentence-BERT: Sentence embeddings using Siamese BERT-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3982–3992, 2019. doi: 10.18653/v1/D19-1410. URL <https://aclanthology.org/D19-1410/>.
- Sentence Transformers contributors. all-MiniLM-L6-v2. Model card, 2021. URL <https://huggingface.co/sentence-transformers/all-MiniLM-L6-v2>. Apache 2.0. Accessed September 2026.
- Ray Smith. An overview of the Tesseract OCR engine. In *Proceedings of the Ninth International Conference on Document Analysis and Recognition (ICDAR)*, volume 2, pages 629–633. IEEE Computer Society, 2007. doi: 10.1109/ICDAR.2007.4376991.
- Ray Smith, Daria Antonova, and Dar-Shyang Lee. Adapting the Tesseract open source OCR engine for multilingual OCR. In *Proceedings of the International Workshop on Multilingual OCR (MOCR)*, pages 1–8. ACM, 2009. doi: 10.1145/1577802.1577804.
- Alan Thomas, Robert Gaizauskas, and Haiping Lu. Leveraging LLMs for post-OCR correction of historical newspapers. In *Proceedings of the Third Workshop on Language Technologies for Historical and Ancient Languages (LT4HALA) @ LREC-COLING 2024*, pages 116–121, 2024. URL <https://aclanthology.org/2024.lt4hala-1.14/>.
- Unicode Consortium. Unicode standard annex #24: Unicode script property. Technical report, The Unicode Consortium, 2024. URL <https://www.unicode.org/reports/tr24/>.
- Wenhui Wang, Furu Wei, Li Dong, Hangbo Bao, Nan Yang, and Ming Zhou. MiniLM: Deep self-attention distillation for task-agnostic compression of pre-trained transformers. In *Advances in Neural Information Processing Systems*, volume 33, pages 5776–5788, 2020.
- Ori Yoran, Tomer Wolfson, Ori Ram, and Jonathan Berant. Making retrieval-augmented language models robust to irrelevant context. In *Proceedings of the Twelfth International Conference on Learning Representations (ICLR)*, 2024. URL <https://openreview.net/forum?id=ZS4m74kZpH>.