GUL'S NOTEBOOK · AI RESEARCH · OPEN ACCESS

RAG over textbooks with tables, formulas and Urdu text

Muhammad Gulzaib
https://orcid.org/0009-0000-8709-6989
SynergyFlow Labs · Islamabad, Pakistan

Published	2026-07-16
URL	https://gulzaib.vercel.app/blog/rag-over-textbooks-with-tables-formulas-and-urdu-text
Licence	Creative Commons Attribution 4.0 International (CC BY 4.0), https://creativecommons.org/licenses/by/4.0/

ABSTRACT

For textbook RAG, extract and review page structure before splitting text into chunks. Keep table headers with rows, equations with definitions, and source page references with each chunk. Test Urdu and mixed-direction text against the original page, then compare retrieval on a labelled question set before trusting generated answers.

Keywords: rag, retrieval, evaluation, edtech, llm

This guide was prepared in November 2026. The displayed 16 July 2026 date was requested for the post; it is not its original publication date.

This is a proposed workflow, not a report of a measured production deployment. The examples and retrieval results below are fictional. No actual syllabus PDF, OCR output or question set was supplied for this article.

Why a page is more than a string

A textbook page can contain a paragraph, a table, a formula and an exercise side by side. A fixed character window may put a table's heading in one chunk and its values in another. It can split an equation from the definitions of its symbols, or join text from two columns in the wrong order.

Retrieval-augmented generation (RAG) searches a source collection and passes selected passages to a model. If the passage is incomplete, the model can produce a fluent answer that the retrieved evidence does not support. The first job is to make sure the index contains usable evidence.

1. Inspect and extract page by page

Start with four checks before indexing a textbook:

1. Page text and reading order.
2. Table headers and rows.
3. Formulas and units.
4. Urdu and mixed direction.

Keep the original PDF and a rendered page image available for review. Record the book, edition, PDF page number, printed page label if present, and location on the page. PDF page 12 and the printed page numbered 12 may be different pages. Check that you have permission to process and store the book and to send its contents to any outside service.

For born-digital PDFs, inspect extracted text and positional blocks. A PDF's internal text sequence may differ from the visual reading order. PyMuPDF documents block and word extraction with coordinates and a `sort` option; top-left sorting is a useful heuristic, not a guarantee for complex layouts or right-to-left columns. [1]

For a scanned page, use optical character recognition (OCR) and compare the result to the image. A selectable text layer can also be wrong, so inspect samples even when text exists. OCRmyPDF supports multiple Tesseract language packs; Urdu uses `urd` and English uses `eng`. Install both trained-data files before using an `eng+urd` configuration. [2] [3] Test Nastaliq pages, low-resolution scans and mixed scripts on the actual material rather than assuming a language pack is accurate enough.

Route pages with missing or suspicious text to review. OCR confidence, where available, is a useful triage signal; it does not prove a formula or table is correct. Keep extractor and OCR versions with each ingest so an index can be reproduced.

2. Keep each table's meaning together

Consider this invented table:

Quantity	Symbol	Unit
Speed	v	m/s
Distance	d	m

A chunk containing `Speed | v | m/s` alone loses the column labels. For a small table, keep its caption, headers and rows together. For a long table, split by rows and repeat the relevant headers, caption and units with each group. Preserve row order and footnotes, including qualifiers that change a value's meaning.

Table extractors can identify regions and return cells, but the result still needs checking: merged cells, borderless tables and tables continuing onto another page are easy to misread. PyMuPDF's `find_tables()` is one option for suitable PDFs. [1] Avoid indexing the same table twice as both loose page text and a structured table without a deduplication or

ranking plan. When extraction is uncertain, keep a link to the original region and flag it for review instead of presenting a guessed table as authoritative.

3. Treat equations as structured evidence

A mathematical expression needs its symbols and surrounding explanation. For example, the invented passage "speed = distance / time, where distance is in metres and time is in seconds" should stay with the equation and its units. If it is too large for one chunk, retain links between the equation and its explanatory blocks and retrieve them together.

Preserve subscripts, superscripts, minus signs, fractions and units. An OCR result that drops an exponent or changes a minus sign changes the answer. Store a faithful text or LaTeX representation when it can be reviewed, alongside the original page and bounding box. If the expression cannot be read reliably, mark the extraction uncertain and use the page image for manual inspection. Do not silently replace unreadable symbols with plausible guesses.

4. Store Urdu in logical order

Urdu uses a right-to-left script, but the same paragraph may contain English terms, numbers or an equation. Unicode distinguishes the logical order of stored characters from their visual display order. [4] Do not reverse strings as a general fix: inspect both the stored text and a correctly rendered view against the PDF, including punctuation and numerals.

Keep the original extracted text, then create a separately versioned search representation if normalisation is necessary. Check character variants and joining marks with language-aware reviewers before applying substitutions; aggressive normalisation can change meaning. For a mixed Urdu-English query, compare lexical and multilingual embedding retrieval on examples from the intended curriculum. Translation and transliteration can be useful query expansions, but they can also change meaning and need their own evaluation.

Make chunks traceable

Split first by document structure: book, chapter and section. Then split long prose at paragraph or sentence boundaries using the embedding model's tokenizer and context limits. A starting experiment might target roughly 300 tokens with modest overlap for prose, but it is a tuning choice, not a measured optimum for every language or book. Do not cut a table row or equation just to meet that target. Handle a block larger than the model limit explicitly, for example by row groups with repeated headers or linked equation and definition chunks.

A useful record carries the source version, grade, subject, chapter, page identifiers, block type, bounding box, extracted text, language, and parser version. Preserve a stable

mapping from chunks back to original source regions. For cross-page tables or worked examples, store all relevant page references rather than attaching the entire chunk to one page.

Apply curriculum and access filters before retrieval, then compare keyword search, multilingual embeddings and a hybrid approach on the same questions. Exact symbols and terms matter in one query; a paraphrase may matter in another. Return the supporting page and region with the answer so a teacher can inspect it. A citation is a pointer, not proof that the answer follows the source.

Measure retrieval separately from answers

Build a question set from material you are allowed to use. A reviewer should label the source regions needed to answer each question, including table headers and units where relevant. Include Urdu, English and mixed-language questions, equations, multi-page evidence and unanswerable questions. Separate a development set from a held-out set, and keep the same queries, corpus and filters when comparing approaches.

For each question, map retrieved chunks to reviewed source evidence. **Any-evidence hit@k** asks whether the top k chunks contain at least one required evidence region. **All-evidence coverage@k** asks whether their union contains every required region. The second measure catches a table row returned without its header. Both are question-level rates, not token recall; unrelated text in a large chunk can inflate coverage, so inspect chunk size and precision as well. Chroma's chunking evaluation discusses the effect of chunking and token-level precision and recall. [5]

A small, runnable scoring fixture

Download the Python scoring script and the JSON fixture into the same directory, then run:

`terminal`

```
python3 textbook-rag-eval.py textbook-rag-eval-fixture.json
```

`fictional fixture output`

```
Fixture evidence coverage at k=3
slice      n   baseline any/all   candidate any/all
overall    3   2/3   1/3          3/3   3/3
table      1   1/1   0/1          1/1   1/1
formula    1   0/1   0/1          1/1   1/1
urdu       1   1/1   1/1          1/1   1/1
```

The baseline and candidate results are hand-written evidence IDs, not output from a retriever. The table case illustrates the distinction: a row appears in the baseline top three,

but the required header does not. In the candidate fixture both appear together. These numbers test the scoring example only; they say nothing about performance on textbooks.

To use this with real results, create stable IDs for reviewed evidence regions in the same source edition, export ranked chunks from each system, and map each chunk to evidence it actually contains. In the JSON, each inner list represents one ranked chunk; an empty list means that chunk contains none of the labelled evidence. The script unions the first k lists. It assumes all listed regions are required, so adapt the labels and scorer for valid alternative sources or partial spans. Review the mapping itself against page images. The fixture has no unanswerable questions; score those separately for false retrieval and appropriate abstention.

Record per-slice counts and inspect individual regressions, including cases where extraction already failed before retrieval began. A small set cannot establish a reliable rate. On a larger held-out set, report its size and uncertainty, index size, latency and context cost alongside quality. Only after the evidence is retrieved should you evaluate whether the final answer is correct, grounded, and appropriately declines when evidence is missing.

*[Figure: image]***Grounded exam generation with RAG: retrieval and evaluation**: How ZeroExam AI keeps generated exam questions inside the syllabus: hybrid retrieval, rank fusion, grounding checks and an evaluation plan you can run. *[Figure: image]***The evals I run before shipping a prompt change**: A practical prompt regression testing workflow: compare baseline and candidate, check hard failures, review quality and gate a release.

> **Key takeaways**
>
> - Inspect extraction against page images before indexing.
> - Keep table headers and formula definitions with the content they explain.
> - Check Urdu in logical storage order and in a rendered mixed-direction view.
> - Track chunks back to specific source pages and regions.
> - Compare retrieval with reviewed evidence and assess generated answers separately.

Questions readers ask

How should I chunk PDF textbooks for RAG?

Extract page structure first, then split at chapter and section boundaries. Keep each table's headers with its rows, preserve formulas with their definitions and retain source page metadata for review.

Does Urdu in a PDF always need OCR?

No. First inspect whether the PDF contains usable, correctly ordered Unicode text. Scanned pages or broken text layers may need OCR and review against the page image.

How do I evaluate textbook retrieval?

Prepare questions with reviewed source evidence and compare systems on the same held-out queries. Measure whether the retrieved context contains all necessary evidence, inspect failures by content type and separately review generated answers.

References

1. PyMuPDF. Text extraction recipes.
2. OCRmyPDF. Installing additional language packs.
3. Tesseract OCR. Tesseract language data.
4. Unicode Consortium. Unicode Bidirectional Algorithm.
5. Smith, B. and Troynikov, A. Evaluating Chunking Strategies for Retrieval, 2024.

CITE AS

Gulzaib, M. (2026, July 16). RAG over textbooks with tables, formulas and Urdu text. Gul's Notebook. https://gulzaib.vercel.app/blog/rag-over-textbooks-with-tables-formulas-and-urdu-text

You may share and adapt this text for any purpose, including commercially, as long as you credit the author and link to the original. Licence: https://creativecommons.org/licenses/by/4.0/