GUL'S NOTEBOOK · AI RESEARCH · OPEN ACCESS

Grounded exam generation with RAG: retrieval and evaluation

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

Published	2026-07-18
URL	https://gulzaib.vercel.app/blog/grounded-exam-generation-with-rag-retrieval-and-evaluation
Licence	Creative Commons Attribution 4.0 International (CC BY 4.0), https://creativecommons.org/licenses/by/4.0/

ABSTRACT

Teachers don't want *creative* exam questions. They want questions that match the syllabus they actually taught. This post describes the retrieval-augmented generation design behind ZeroExam AI: hybrid lexical and dense retrieval over syllabus chapters, reciprocal rank fusion, a grounding check on every generated question, and an evaluation plan that separates retrieval quality from generation quality.

Keywords: rag, llm, evaluation, retrieval, edtech

The problem

A plain LLM asked to "write ten physics questions for grade 9, chapter 3" produces fluent questions, but some of them test concepts that aren't in that chapter, or in that country's curriculum at all. For a teacher, one off-syllabus question is enough to stop trusting the tool.

RAG is the obvious fix: retrieve the relevant chapter content and make the model generate from it [1]. The details are where it gets interesting.

Pipeline

1. **Chunk**: by heading, ~300 tokens
2. **Retrieve**: BM25 + dense, k = 20 each
3. **Fuse**: reciprocal rank fusion
4. **Generate**: JSON schema, cite chunk IDs
5. **Verify**: is each question grounded?

Figure 1. Every generated question carries the IDs of the chunks it came from.

Chunking by structure

Textbooks already have structure: chapters, sections, worked examples, exercises. I chunk along headings first and only split long sections by token count. Each chunk keeps its metadata (`grade`, `subject`, `chapter`, `section`), which lets us filter *before* ranking. Filtering by chapter removes most of the off-syllabus risk on its own.

Hybrid retrieval

Dense embeddings are good at paraphrase ("speed" and "velocity"). Lexical search like BM25 [2] is good at exact terms, formulas and names that embeddings blur together. Exam content has plenty of both, so I run both and fuse the rankings.

Reciprocal rank fusion

Scores from BM25 and cosine similarity live on different scales, so instead of mixing scores, I mix *ranks*. Reciprocal rank fusion [3] scores a document d across the ranked lists R as:

$$\mathrm{RRF}(d) = \sum_{r \in R} \frac{1}{k + \mathrm{rank}_r(d)}$$

where k is a smoothing constant, commonly $k = 60$. A chunk that appears near the top of either list gets a boost, and one that appears in both does best. It needs no tuning and no score normalisation, which is exactly what you want in an early system.

`fusion.py`

```
def rrf(rankings: list[list[str]], k: int = 60) -> list[str]:
    scores: dict[str, float] = {}
    for ranking in rankings:
        for rank, doc_id in enumerate(ranking, start=1):
            scores[doc_id] = scores.get(doc_id, 0.0) + 1.0 / (k + rank)
    return sorted(scores, key=scores.get, reverse=True)
```

Context ordering

Models use information at the start and end of a long context more reliably than information in the middle [4]. So I keep the context short (top 6–8 chunks after fusion) and put the strongest chunks first.

> **Insight**
>
> More context is not better context. Fewer, well-ranked chunks gave cleaner questions than stuffing the whole chapter into the prompt.

Grounding every question

Generation returns structured JSON. Each question must list the chunk IDs that support it:

`question.json`

```
{
  "question": "A car accelerates uniformly from rest to 20 m/s in 5 s. Find its
acceleration.",
  "type": "numerical",
  "difficulty": "easy",
  "answer": "4 m/s²",
  "sources": ["phy-9-ch3-s2-c04"]
}
```

A second, cheaper model call then acts as a verifier: given only the cited chunks, *can this question be answered, and is the answer correct?* Questions that fail are regenerated or dropped. This catches most hallucinated concepts before a teacher ever sees them.

Evaluation plan

The most common mistake I see in RAG projects is evaluating only the final answer. If the output is bad, you can't tell whether retrieval or generation failed. So I measure them separately, in the spirit of RAGAS [5].

Stage	Metric	Definition	Target
Retrieval	Recall@k	Share of relevant chunks found in the top k	High at $k = 8$
Retrieval	MRR	Mean of $1/\text{rank}$ of the first relevant chunk	Above 0.5
Generation	Faithfulness	Share of questions fully supported by cited chunks	Close to 100%
Generation	Syllabus match	Teacher rates question as in-syllabus (yes/no)	Close to 100%
Product	Edit rate	Share of questions a teacher edits before printing	Trending down

Recall at k for a query with relevant set G and retrieved list $\hat{R}_k$:

$$\mathrm{Recall@}k = \frac{\lvert G \cap \hat{R}_k \rvert}{\lvert G \rvert}$$

> **Build the gold set with teachers**
>
> Ask two or three teachers to write 30 realistic requests and mark which sections answer them. It takes an afternoon, and it turns "it seems better" into a number.

Latency budget

A teacher will wait a few seconds for a full paper, but not a minute. This is the budget I design against for a single question, not a benchmark:

Design budget per question (ms)

Filter + BM2540 ms

Embed query + ANN120 ms

Fusion5 ms

Generation1800 ms

Verification600 ms

Generation dominates, so questions for a paper are generated in parallel and streamed into the editor as they're ready.

Open questions

Things I'm still exploring

- **Answer checking.** Grading handwritten or typed long answers against a rubric is a much harder problem than generating questions.
- **Urdu-medium content.** Tokenisation and embeddings for Urdu are weaker, which affects both retrieval and generation.
- **Difficulty calibration.** "Hard" according to the model doesn't always mean hard for students. Real student results are the only reliable signal.

> **Key takeaways**
>
> - Filter by metadata first. It removes most off-syllabus risk cheaply.
> - Use hybrid retrieval and fuse ranks, not scores.
> - Make every generated item cite its sources, then verify it.
> - Evaluate retrieval and generation separately.

Questions readers ask

How do you stop RAG from generating off-syllabus questions?

Filter retrieval by metadata such as grade, subject and chapter first, retrieve with hybrid lexical and dense search, and make every generated question cite its source passages so a grounding check can verify it.

Why use reciprocal rank fusion for hybrid retrieval?

Lexical and dense retrievers produce scores on different scales. Fusing their ranks instead of their scores combines them without fragile score normalisation.

How should a RAG system be evaluated?

Evaluate retrieval and generation separately, so you know whether a bad question came from missing context or from the model.

References

1. Lewis, P. et al. Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. NeurIPS, 2020. https://doi.org/10.48550/arXiv.2005.11401
2. Robertson, S. and Zaragoza, H. The Probabilistic Relevance Framework: BM25 and Beyond. Foundations and Trends in Information Retrieval, 2009. https://doi.org/10.1561/1500000019
3. Cormack, G. V., Clarke, C. L. A. and Büttcher, S. Reciprocal Rank Fusion Outperforms Condorcet and Individual Rank Learning Methods. SIGIR, 2009. https://doi.org/10.1145/1571941.1572114
4. Liu, N. F. et al. Lost in the Middle: How Language Models Use Long Contexts. TACL, 2024. https://doi.org/10.1162/tacl_a_00638
5. Es, S. et al. RAGAS: Automated Evaluation of Retrieval Augmented Generation, 2023. https://doi.org/10.48550/arXiv.2309.15217

CITE AS

Gulzaib, M. (2026, July 18). Grounded exam generation with RAG: retrieval and evaluation. Gul's Notebook. https://gulzaib.vercel.app/blog/grounded-exam-generation-with-rag-retrieval-and-evaluation

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/