

A Retrieval-Augmented Generation Pipeline for Enterprise-Tier Documents: Architecture, Implementation, and Behaviour Analysis

Jomari Manto
College of Computer Studies & Engineering
Jose Rizal University
Mandaluyong City, Philippines
jomari.manto@my.jru.edu

Abstract—Retrieval-Augmented Generation is an architecture that pairs a language model with an external document store, so that answers are drawn from a defined body of source material rather than from the model’s own training. This paper examines how such a pipeline behaves in practice. A six-stage implementation was built and instrumented: document loading, chunking, embedding with all-MiniLM-L6-v2, indexing in ChromaDB, fixed top-k retrieval, and constrained generation through a hosted model. Behaviour was measured across in-scope, near-domain, and out-of-domain queries under two prompt and temperature configurations, using an enterprise-tier corpus as the test case. Hallucination occurred with all prompt safeguards active, and both instances originated in what retrieval supplied rather than in how generation was configured. Removing the safeguards altered response wording more than behaviour. Retrieval quality, not prompt constraint, determined whether the pipeline stayed grounded.

Keywords—retrieval-augmented generation, vector embeddings, semantic search, context grounding, chunking, LLMops

I. INTRODUCTION

Natural language processing has advanced through several architectural generations, each addressing a limitation of the one before it. Early statistical methods depended on hand-engineered features. Neural approaches replaced these with learned word embeddings, and recurrent architectures such as LSTM and BiLSTM added the ability to model sequence context in both directions. The attention mechanism followed, allowing a model to weight the parts of an input that matter most to a given prediction instead of treating all positions equally [1]. Transformer architectures generalised attention into the primary computation [2] and scaling those architectures across very large corpora produced the present generation of large language models.

A model of this kind is a *next-token predictor*. Starting from an initial prompt, it generates one token at a time by sampling from a probability distribution over the vocabulary, appends that token to the sequence, and uses the extended sequence as context for the token that follows. Repeated, this autoregressive loop produces coherent passages. It is also the source of the central weakness: a next-token predictor optimises for what is likely to follow, not for what is true, so the process offers no guarantee that the resulting statement is factually correct. Output that is fluent, confident, and unsupported by any real source is termed hallucination. It is

difficult to detect precisely because it carries no signal of uncertainty: an invented figure is presented in the same register as a correct one, so a reader without access to the source material has no basis for distinguishing them.

The problem is more acute for enterprise-tier data. Foundational knowledge is broad public text absorbed during pretraining. Domain knowledge covers a specialised but public field. Enterprise knowledge belongs to a single organisation, and a foundation model has never encountered it and could not have. Fine-tuning can inject such knowledge but is costly and must be repeated whenever the source changes.

Retrieval-Augmented Generation takes a different approach. Rather than modifying the model, it retrieves relevant passages from an external store at query time and places them in the model’s context window [3]. Surveys of the field describe this as the dominant mitigation strategy for knowledge-intensive tasks [4]. The distinction from the training paradigm is fundamental, as summarised in Table I.

TABLE I. NEURAL NETWORK TRAINING COMPARED WITH RETRIEVAL-AUGMENTED GENERATION

Aspect	Training	RAG
What changes	Model weights, via backpropagation	Nothing; weights remain frozen
Knowledge location	Encoded into parameters	Supplied in the context window at query time
Cost to update	Full or partial retraining	Re-index the corpus, minutes
Failure modes	Overfitting, underfitting	Hallucination, retrieval miss
Performance measure	Training and validation loss	Retrieval hit rate, grounding accuracy
Auditability	Opaque; knowledge distributed across weights	Traceable; retrieved passages inspectable

Because no gradient is computed, a RAG pipeline has no loss curve. Its performance must be assessed by whether retrieval supplies the passage containing the answer, and whether the model uses that passage faithfully. This paper reports the architecture, implementation, and observed behaviour of such a pipeline, and examines whether the safeguards commonly assumed to prevent hallucination do so in practice.

II. SYSTEM ARCHITECTURE

The pipeline consists of six stages executed in two phases. Indexing runs once, when the corpus is prepared. Retrieval and generation run on every query.

Document loading. The source corpus [5] is read page by page by PyPDFLoader, producing 42 documents, each carrying extracted text together with metadata identifying the source file and page number. This metadata survives every later stage and is what allows an answer to be traced back to a specific page.

Chunking. Page-length documents are too coarse for retrieval, since a whole page dilutes a relevant sentence with unrelated text. RecursiveCharacterTextSplitter divides each page into segments of roughly 500 characters, attempting paragraph breaks first, then line breaks, then spaces, so that cuts fall at the largest natural boundary within the size limit. A 50-character overlap protects statements that straddle a boundary from being severed.

Embedding. Keyword matching cannot relate a question about permitted absences to text describing a twenty percent threshold, since the two share no vocabulary. Embedding maps text into a vector space where semantic similarity corresponds to proximity. The all-MiniLM-L6-v2 sentence transformer [6], [7] produces a 384-dimension vector per chunk and executes locally, so no part of the corpus is transmitted externally during indexing.

Vector store. ChromaDB [8] stores each vector alongside its originating text and metadata. At query time it embeds the incoming question using the same model and returns the chunks whose vectors lie nearest.

Retrieval. The retriever returns the three nearest chunks. It applies no relevance threshold, so three chunks are returned for every query regardless of quality, and the retriever has no mechanism for reporting that nothing suitable was found. This property is central to the behaviour reported in Section IV.

Generation. The retrieved chunks and the question are inserted into a prompt template carrying three standing instructions: an identity, a restriction limiting the answer to the supplied context, and explicit wording for refusing when the context is insufficient. The assembled prompt is sent to a model hosted on Groq. The response returns both the generated answer and the retrieved passages, which makes the output auditable.

III. IMPLEMENTATION DETAILS

TABLE II. PIPELINE CONFIGURATION

Parameter	Value
Chunk size	500 characters
Chunk overlap	50 characters
Embedding model	all-MiniLM-L6-v2
Vector dimensions	384
Vector store	ChromaDB, in-memory
Top-k retrieval	3
Relevance threshold	None
Generation model	openai/gpt-oss-20b (Groq)
Temperature, baseline	0
Temperature, stress-test	1.0
Corpus	42 pages, 269 chunks

The pipeline was assembled using the LangChain framework [9], which supplies the loader, splitter, vector-store wrapper, and chain abstractions described above. Three implementation deviations from the supplied template were necessary, and each is an operational observation rather than a coding matter.

The template specified llama-3.1-8b-instant. Groq deprecated this model on 17 June 2026 and withdrew it on 16 August 2026 [10], so requests returned a 404 model_not_found error. The substitute, openai/gpt-oss-20b, is the replacement documented by the provider. No code changed: a hardcoded model identifier in a document written some months earlier had become a broken pipeline, and the failure surfaced only at the first live API call.

Two further breakages had the same character. LangChain 1.0 relocated the chain constructors out of langchain.chains into a compatibility package, so the template imports no longer resolved; and the DirectoryLoader configuration fell back to a parser whose PDF dependencies were absent, raising an ImportError that prevented the corpus from being read at all. An unpinned dependency stack breaks when upstream components change, and the failure presents as a missing module or an unknown model rather than as anything resembling the cause. Chunking parameters, retrieval depth, temperature, and prompt wording were left at their template values throughout, so the behaviour reported below reflects the configuration as specified.

IV. EXPERIMENTAL RESULTS AND BEHAVIOUR ANALYSIS

Behaviour was examined using four queries under two configurations. The baseline used temperature 0 with the refusal instruction present. The ungrounded configuration raised temperature to 1.0 and deleted the refusal instruction, with retrieval held constant so that any difference was attributable to the model and prompt alone. Each ungrounded query was run three times, since output is non-deterministic at temperature 1.0. Table III presents the test queries, expected responses, and actual model outputs required for this analysis. A larger evaluation, reported in the final part of this section, tests whether the resulting pattern holds across a wider question set.

TABLE III. QUERY RESULTS ACROSS BOTH CONFIGURATIONS

Query	Expected	Baseline	Ungrounded
In-scope summary	Grounded	Hallucinated	Not run
A. Visa requirements	Refusal	Refused, verbatim	Refused ×3
B. Tuition per unit	Refusal	Refused, verbatim	Refused ×3
C. Grade, other institution	Refusal	Hallucinated	Hallucinated ×2, refused ×1

Hallucination under the baseline. The first in-scope query returned descriptions of four sections the model had never received. Two of its three retrieved chunks were table-of-contents pages containing section titles and page numbers but no body text, and the model produced plausible descriptions of what each titled section would contain. This occurred at temperature 0 with the refusal instruction active, that is, under the configuration treated as the control.

Retrieved content determines behaviour more than topical distance. The three unanswerable queries received different kinds of context and produced correspondingly different outcomes. Query A retrieved passages sharing only superficial structure with the question and was refused. Query B retrieved

the exact section discussing the topic, which is topically correct but states no figure, and was also refused. Query C retrieved criteria containing a numeric threshold and was answered. Query B is the informative control: topical relevance alone did not induce hallucination. Query C did so because the retrieved context contained a fragment resembling the requested answer. The same mechanism explains the baseline case, where section titles resembled section contents.

Grounded but false answers. The value returned for Query C appears in the corpus as an honours eligibility threshold, not as a passing grade, and belongs to a different institution than the one asked about. Every element of the answer traces to real source text, and the answer is false. Verification by citation would not detect this, because the citation is valid; the interpretation is not.

Removing the safeguards altered wording more than behaviour. Across nine ungrounded calls, no query that refused under the baseline was converted into a hallucination, and the query that hallucinated continued to do so in two of three runs. What changed was consistency. Under the baseline the prompt supplied exact refusal wording, which the model reproduced identically. With that instruction removed, six refusals produced six distinct strings expressing the same judgement. A refusal rate cannot be reported if refusals cannot be identified programmatically, so the instruction was performing two functions, and the standardising one proved more consequential.

Temperature affects predictability rather than correctness. At temperature 1.0 the same query produced a confident hallucination, a hedged hallucination, and a refusal across three identical calls. The baseline response, by contrast, was reproduced character for character across two separate runtimes with independently rebuilt indexes.

Architectural dependencies. The two models carry different risk profiles. The embedding model executes locally and remained available throughout; the generation model is a hosted service and was withdrawn mid-project. Hosted inference introduces an operational dependency that local inference does not, and the asymmetry is a property of the architecture rather than of the corpus.

Extended evaluation. The four queries above establish a pattern but cannot show whether it holds generally. A larger evaluation was therefore constructed: 45 questions in three categories, comprising 15 answerable from the corpus, 15 near-domain questions whose topic the corpus discusses without stating the specific answer, and 15 unrelated to it. For each answerable question the page containing the answer was located in the source beforehand, so retrieval could be scored by comparing retrieved page numbers against ground truth. That comparison requires no call to the language model, which allowed a parameter search to run at local speed; the model was contacted only for the final validation.

The 15 answerable questions were split before the search began, 10 used to select the configuration and 5 held back. Tuning against the same questions used to report the final score would yield a figure reflecting settings chosen to suit those questions, so the held-out set provides a measure that was not optimised for. The search varied table-of-contents filtering, then chunk size, then retrieval depth, stopping at a 90 percent hit rate or twelve configurations.

The search terminated after two configurations. The template settings reached 90 percent at the first measurement, so chunk size and retrieval depth were never tested. Filtering table-of-contents chunks removed 33 of 269 segments and

improved mean reciprocal rank from 0.767 to 0.800 with the hit rate unchanged. This is a marginal gain in ranking, not a correction: the pipeline was not tuned up to 90 percent, it began there.

TABLE IV. EXTENDED EVALUATION RESULTS

Measure	Result
Retrieval hit rate, tuning set	9 of 10
Retrieval hit rate, held-out set	5 of 5
Mean reciprocal rank, held-out set	0.800
Correct refusals, far-domain	15 of 15
Correct refusals, near-domain	14 of 15
Correct refusals, overall	29 of 30

The single refusal failure repeats the mechanism identified above. Asked the amount of an identification replacement fee, the pipeline answered with a specific peso figure. The corpus mentions such a fee but never states its value; the figure returned is a processing charge drawn from an unrelated clause on withdrawals, and is the only monetary amount appearing anywhere in the 42 pages. The question requested a number, the retrieved context contained a number, and the model supplied it. This reproduces the Query C result on a question set constructed for a different purpose, which is the strongest evidence available here that the behaviour is systematic rather than incidental.

One retrieval miss is also informative. The question on disciplinary sanctions failed under both configurations, retrieving adjacent pages instead of the correct one. That page holds a sanctions table laid out as a grid; the splitter reads it as a character stream, so its structure is lost and its cells are interleaved with surrounding text. Tabular content appears systematically harder to retrieve than prose.

Two qualifications apply. The answerable questions were written by selecting facts stated unambiguously in one place, which makes the retrieval task easier than an arbitrary question would be, so the hit rate is an upper bound rather than an estimate of general performance. More importantly, the search space contained no control addressing the mechanism that produces the failures. Chunk size and retrieval depth adjust how much text is returned; neither can reject a poor match. A relevance threshold, which would return an empty context when no chunk scores above a minimum similarity, was not among the parameters available to vary.

V. CONCLUSION

What the architecture solves. Earlier approaches to grounding a language model in specific knowledge each carry a structural cost. Prompt engineering alone cannot introduce information the model does not already hold. Fine-tuning can, but it is expensive, must be repeated whenever the source changes, and leaves the injected knowledge distributed opaquely across weights, so a wrong answer cannot be traced to a wrong source. Retrieval-Augmented Generation avoids both. Knowledge enters through the context window rather than the parameters, so a corpus can be replaced by re-indexing in minutes, and every answer arrives with the passages that produced it. That last property is what made the analysis in Section IV possible at all: the hallucinations reported there were identified by reading the retrieved chunks against the source, which no fine-tuned model would permit.

What it does not solve. Retrieval reduces hallucination; it does not eliminate it. Both hallucinations in the four-query set

and the one in the extended evaluation occurred with the retrieval layer functioning exactly as designed. The architecture constrains where the model may look. It does not constrain what the model concludes from what it finds, and the evidence here is that a fragment bearing surface resemblance to the requested answer is sufficient to produce a confident and false response. Prompt restrictions cannot close this gap, since they are conditioned on the context being empty rather than on its being uninformative; the model's own judgement of sufficiency is what decides, and that judgement is satisfied by resemblance. Temperature affects only how consistently the same judgement is expressed. Protecting the API credential, while necessary, is an operational matter and bears on account security rather than on answer quality.

Dependence on the generation model. A RAG pipeline is not model-agnostic in practice, even though it is in principle. The retrieved passages are identical regardless of which model receives them, but the interpretation is not. The failure reported in Section IV turned on a numeric threshold being read as a passing grade when the corpus defines it as an honours criterion, a distinction that depends on familiarity with the conventions of the field rather than on anything present in the retrieved text. A model whose pretraining covered the relevant domain would be better placed to recognise that mismatch. Enterprise-tier content cannot be pretrained, by definition, but the domain surrounding it can be, and model selection should weigh that rather than treating the generation stage as interchangeable. A pipeline built on identical data and parameters may behave differently under another model; this was not tested here, since the template model was unavailable, and is stated as a hypothesis.

Structural weaknesses. Three further limitations are properties of the architecture as commonly implemented rather than of any particular corpus. Retrieval returns a fixed number of results with no relevance threshold, so it cannot report that nothing suitable was found and instead supplies the least unsuitable candidates available. This behaviour follows the probabilistic logic set out in Section I: a next-token predictor selects the likeliest continuation rather than the true one, and retrieval applies the same kind of ranking one stage earlier, returning the nearest chunks whether or not they are near. Neither stage holds any representation of correctness [4]. Chunking operates on character counts and is blind to document structure, so navigational material such as tables of contents is indexed alongside substantive text and competes with it on equal terms, while tabular content is fragmented and becomes systematically harder to retrieve. And no stage verifies that the answer follows from the passages cited, which is why an answer can be traceable to real source text and still be false. Each of these is addressable, by a similarity threshold, by structure-aware preprocessing, and by a verification pass respectively, but none is addressed by the default configuration.

Applications. The architecture suits any setting where an authoritative corpus belongs to a single organisation and changes faster than retraining permits: institutional policy assistants, product support systems, and internal technical documentation. In each case the same caution applies. Grounding a response in a retrieved passage constrains what the model may draw upon; it does not ensure the passage was understood correctly. Where the cost of a confident error is high, retrieval alone is not a sufficient safeguard.

Retrieval-Augmented Generation makes institutional knowledge usable by a model that has never seen it, at a cost measured in minutes of re-indexing rather than in retraining, and it returns every answer with the evidence that produced it. Those two properties are what recommend it wherever an authoritative corpus is private, consequential, and subject to revision. The failures observed here do not contradict that case; they mark where the remaining work lies. Each arose at a point in the pipeline that ranks candidates without assessing them, which is also where the clearest opportunities for improvement sit: a similarity threshold so that retrieval can decline to answer, preprocessing that distinguishes substantive text from navigational structure, a verification stage that tests whether an answer follows from the passage cited, and deliberate selection of a model whose pretraining covers the surrounding domain. Taken together, these results position Retrieval-Augmented Generation as a substantial advance rather than a settled solution.

REFERENCES

- [1] A. Duan and R. C. Raga, "BiLSTM model with attention mechanism for multi-label news text classification," in 2024 4th Int. Conf. Neural Networks, Information and Communication Engineering (NNICE), Guangzhou, China, 2024, pp. 566–569.
- [2] A. Vaswani et al., "Attention is all you need," in Proc. 31st Int. Conf. Neural Information Processing Systems (NeurIPS), 2017, pp. 5998–6008.
- [3] P. Lewis et al., "Retrieval-augmented generation for knowledge-intensive NLP tasks," in Proc. 34th Int. Conf. Neural Information Processing Systems (NeurIPS), 2020, pp. 9459–9474.
- [4] Y. Gao et al., "Retrieval-augmented generation for large language models: A survey," arXiv:2312.10997, 2024.
- [5] Jose Rizal University, Student Handbook SY 2024–2025, College Division. Mandaluyong City, Philippines, 2024.
- [6] N. Reimers and I. Gurevych, "Sentence-BERT: Sentence embeddings using Siamese BERT-networks," in Proc. EMNLP-IJCNLP, 2019, pp. 3982–3992.
- [7] Hugging Face, "sentence-transformers/all-MiniLM-L6-v2 model card." [Online]. Available: <https://huggingface.co/sentence-transformers/all-MiniLM-L6-v2>
- [8] Chroma, "Chroma documentation." [Online]. Available: <https://docs.trychroma.com/>
- [9] LangChain, "LangChain documentation." [Online]. Available: <https://python.langchain.com/docs/>
- [10] Groq, "Deprecations — GroqCloud documentation." [Online]. Available: <https://console.groq.com/docs/deprecations>