Engineer

Retrieval-augmented generation

generating answers from documents retrieved per question

What it is

Retrieval-augmented generation is a way of answering a question with a language model in two steps. First the application searches a collection of documents for the passages most likely to hold the answer. Then it puts those passages into the model's input next to the question, and the model writes its answer from them rather than from what it absorbed during training. The name, usually shortened to RAG, comes from a 2020 research paper on knowledge-intensive language tasks such as answering open questions.

That paper described two kinds of memory. A pre-trained model keeps what it learned in its parameters, which the authors called parametric memory; a separate index of documents, searched by a trained retriever, is non-parametric memory. Their system joined a pre-trained sequence-to-sequence model to a dense vector index of Wikipedia, and they compared two ways of using what came back: one conditions the whole answer on the same retrieved passages, the other may draw on different passages for each token it writes. On three open-domain question-answering benchmarks it set new state-of-the-art results, and the authors found its generated text more specific, more varied and more factual than that of a comparable model without retrieval.

An application can build the same pattern from parts it already has: a search index over its own documents and a large language model it calls over an API. The retrieval half is ordinary search, and both of its families appear here. Sparse retrieval matches words, in the way full-text search ranks documents with a scoring function such as BM25. Dense retrieval turns the question and every passage into vectors of numbers and returns the passages whose vectors lie closest, which is vector search. A 2020 paper on dense passage retrieval found that a dense retriever trained on a modest number of question and passage pairs

beat a strong BM25 baseline by 9 to 19 percentage points on how often the right passage appeared among the first twenty results. A system can run both and merge the two lists.

Use the concept in a decision

Put “What is retrieval-augmented generation?” to work in a real decision. We can help relate the concept to the system you are building or buying. Your records where AI agents can reach them, one version of each customer, nobody shown more than they should, and every answer traced to its source.

Explore our services Get in touch →

Test the assumptions behind the term

Use the explanation and linked work to frame questions for your project. We can examine the assumptions with you before a tool is chosen. Made a geographical data pipeline 50x faster — 50x faster geographical data pipelines

Made a geographical data pipeline 50x faster Get in touch →

Why it matters

A language model knows only what was in its training text, and that text ends on a fixed date and never included your contracts, your price lists or last week’s minutes. The RAG paper names two limits of knowledge kept in parameters: it is hard to show where an answer came from, and hard to update. Retrieval addresses both. The answer can cite the passages it was given, so a reader can open the source and check it, and new knowledge reaches the model the moment a document enters the index, with no training run at all.

It also puts the question of who may see what back where it belongs. Retrieval is a search, and a search can run with the rights of the person asking, so the model is shown only passages that person could have opened themselves. A model trained on the same documents has no such boundary: everything it learned is available to everyone who uses it.

The cost is that the answer can be no better than what the search returns. A model given the wrong passages writes a fluent answer from them, and a model given nothing useful may answer from memory regardless. Most of the work in a retrieval-augmented system therefore goes into the data and the search, not into the model.

What it looks like in practice

The trace below follows one question through a retrieval-augmented system. The search runs with the asking user's permissions, each passage arrives with the file and page it came from, and the instructions tell the model to answer from the passages and nothing else:

```
question: Which delivery terms apply to order 4471?
user:     sales-3 (role: sales, region: north)

retrieved, filtered to what sales-3 may read:
  [1] contracts/framework-2025.pdf, page 4
      "Goods are delivered to the buyer's warehouse
       unless the order states otherwise."
  [2] orders/4471.txt, line 12
      "Delivery terms: as per framework agreement."

instructions:
  Answer only from the passages below and cite them as [n].
  If they do not answer the question, say so.

answer:
  Order 4471 refers to the framework agreement [2], under
  which goods are delivered to the buyer's warehouse [1].
```

Every part of that trace is data the system can check. A test can confirm that the right passages came back for a known question, a script can confirm that every `[n]` in the answer refers to a passage that was actually retrieved, and a person reading the answer can open page 4 of the contract.

The groundwork this needs is the subject of this site's page on company data ready for AI agents. It describes turning old PDFs, scans and mail archives into searchable text with their source kept beside them, and having every row keep where it came from — the system, the load, the file and the page — so an agent can cite the record behind a figure and a person can open it. The page also puts the rules in the data rather than in the prompt: the agent acts with its user's privileges, never an administrator's, and every read is logged with the person it was made for.

Where it goes wrong

The search misses, and the model answers anyway. When the right passage is not retrieved, the model still receives something and still writes an answer. Measure retrieval on its own: write down real questions together with the passages that answer them, and check how often those passages appear among the results the model is given. Tell the model to say so when the passages do not answer the question, and include questions like that among your test cases.

Passages that cut the answer in half. Splitting documents into pieces of a fixed length separates a table from its heading and a clause from its exception, and either half retrieved alone misleads. Split along the structure the document already has, its sections, pages and table rows, and keep the document's title and location with every piece.

The index ignores who is asking. An index built by a service that can read everything will return passages from documents the person asking may not open, and the model will summarise them for that person. Filter by the user's permissions inside the search, in the index or the database, before anything reaches the model. A rule in the prompt is not access control.

Instructions planted in a document. Retrieved text is text somebody else wrote, and the model cannot reliably tell a passage from an instruction. NIST's profile for generative AI describes indirect prompt injection as attacks on applications built on language models through data likely to be retrieved, and a 2023 paper demonstrated such attacks against real systems. Treat every retrieved passage as untrusted input, let no passage trigger an action without a check in code, and give the tools the model can call no more rights than the user has.

An index that falls behind. A document is changed or deleted, and the index still holds the old passages, so the system confidently answers from a contract that no longer applies. Update the index whenever a source changes, remove passages when their document is deleted, and show the date of each source next to the answer.

Citations that do not say what the answer claims. A model can cite passage [2] for a sentence passage [2] does not support, and NIST notes that generated citations can themselves be confabulated. Check in code that every citation points at a passage that was retrieved, and have people sample answers against their cited passages as part of regular testing.

Further reading

- Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks — the 2020 paper that introduced RAG models and compared its two formulations on question answering and generation.
- Dense Passage Retrieval for Open-Domain Question Answering — a dense retriever trained from question and passage pairs, measured against BM25 on finding the passage that holds the answer.
- NIST AI 600-1: Generative Artificial Intelligence Profile — NIST's risk profile for generative AI, covering confabulation, data privacy and indirect prompt injection.
- Not what you've signed up for: Compromising Real-World LLM-Integrated Applications with Indirect Prompt Injection — how instructions hidden in retrieved data take over an application, with attacks on real systems.

Glossary

Choose for your actual constraints

Bring a representative dataset, the questions it must answer and the current bottleneck. We can agree how to measure quality, throughput and recovery.

See what it costs Get in touch →

Make the terminology practical

Ask how "What is retrieval-augmented generation?" applies to your organisation. A concrete example from your current system is enough to start the discussion.

Explore our services

welcome@engineer.company +45 50 10 61 84 Copenhagen, Denmark

Take this page with you

Contact card

Carefully Crafted With Love by D.

CVR 45969703 · Tagensvej 105C, 3. th, 2200 København N

Copyright © 2025 – present · Engineer ApS · All rights reserved.

Printed from https://engineer.company/glossary/retrieval-augmented-generation/