AI Advances · Follow publication

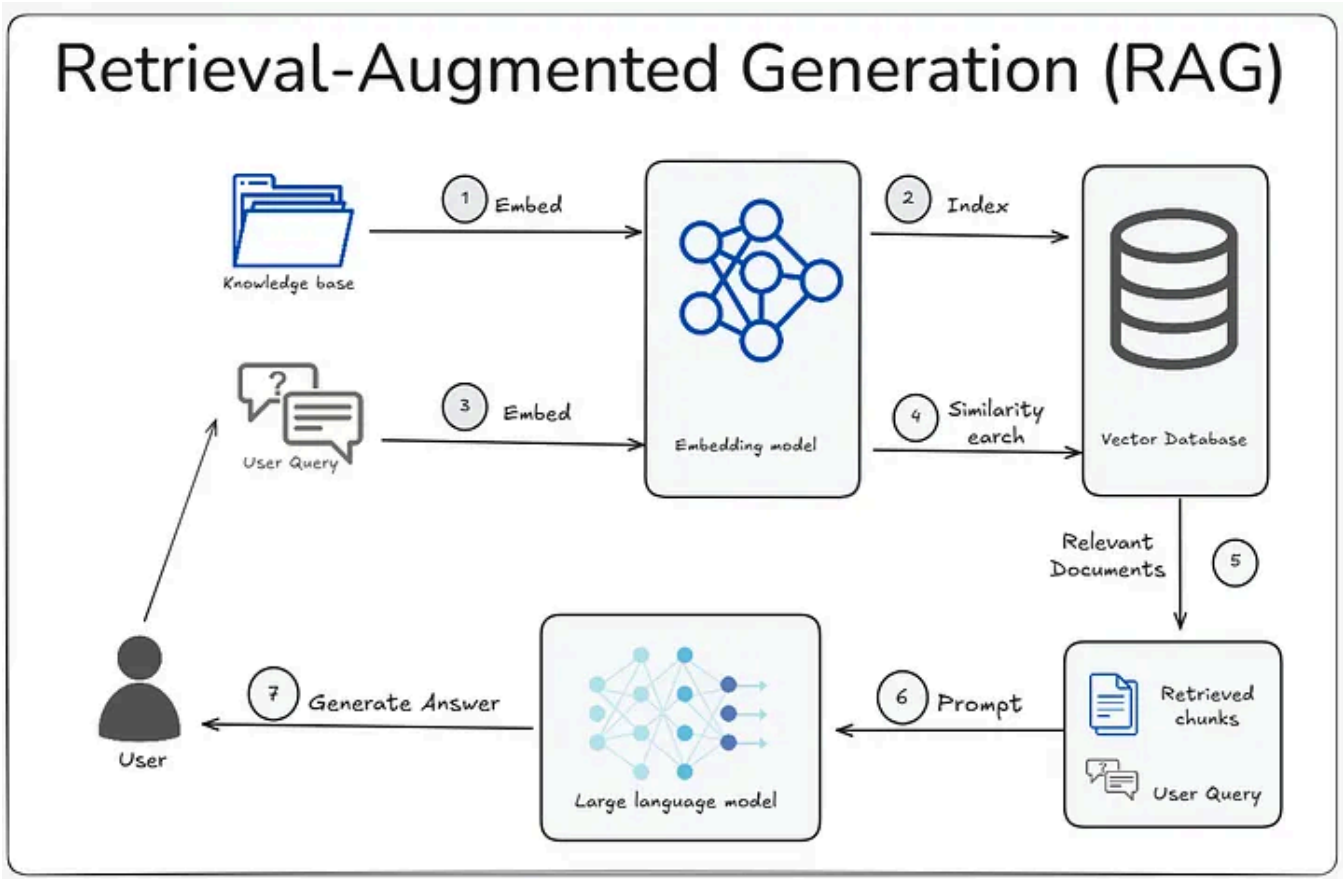
Member-only story

21 Chunking Strategies for RAG

And how to choose the right one for your next LLM application

8 min read · Jun 30, 2025

Anjolaoluwa Ajayi Follow



A Quick Refresher on RAG | Image by Author

Retrieval-Augmented Generation (RAG) is one AI technique many AI engineers hate to love (me included).

Yes, because, on paper, it sounds so simple: "*Retrieve* the right context from your custom data and let an LLM *generate* a response based on it."

But in practice, you're stuck wrangling gigabytes of messy data stored in the most chaotically random formats you've ever seen, then doing days of trial and error:

- tweaking chunks
- switching embedding models
- swapping out retrievers
- fine-tuning rankers
- rewriting prompts

And the model still says, "I don't have enough info to answer your query."

Or worse, it *confidently* churns out total nonsense (hallucinates).

No doubt, there are many moving parts in RAG, but one piece that quietly determines whether the whole thing works or not is **chunking**.

Different data types, file formats, content structures, document lengths, and use cases all call for different chunking strategies.

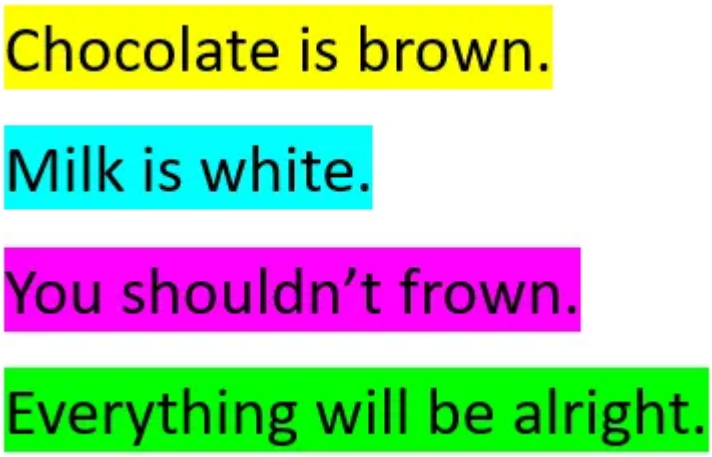
Get it wrong, and your model either misses the point or, well, misses the point…

In this article, we will look at 21 chunking strategies (from easy to advanced) and when to use each one so that your RAG systems stop… missing the point.

Let's Do This GIF

1. Naive chunking (split by newline)

You split the text at every line break. That's it.



Example of Naive chunking | Image by author

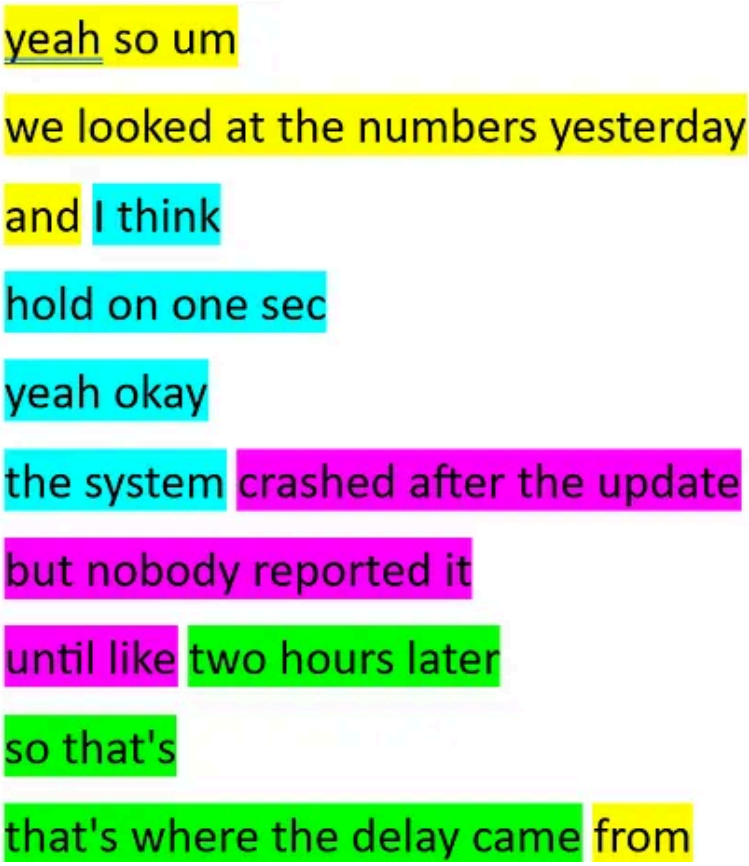
When to use it:

- For text that's *uniformly* separated by newlines: notes, bullet lists, FAQs, chat logs, or transcripts where each line holds a complete thought.

P.S. If lines are too long, you could blow the LLM token limit. If they're too short, the model might miss context or hallucinate.

2. Fixed-size/ fixed window chunking

You break the text into equal parts by word or character count (even if it cuts through a sentence or thought).



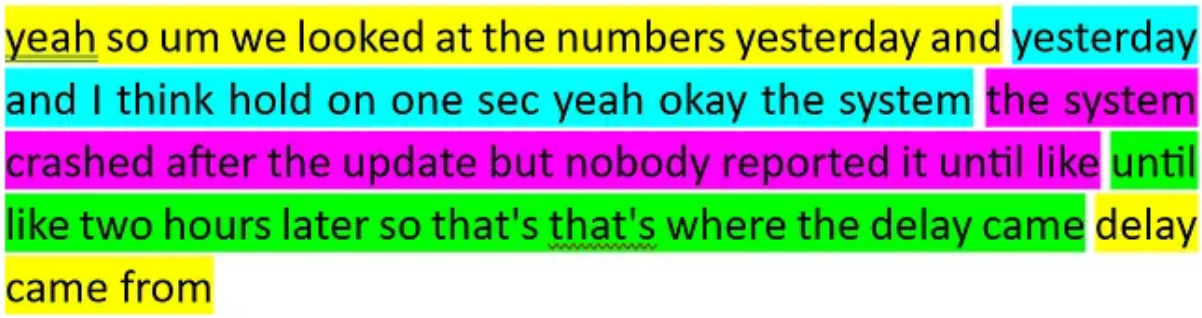
Example of fixed-size chunking | Image by author

When to use it:

- For raw, messy text dumps, such as scanned documents, bad transcripts, or large text files with no punctuation or headings or… structure.

3. Sliding window chunking

Like fixed-size chunking, but each chunk overlaps with the one before it to maintain context across chunks.



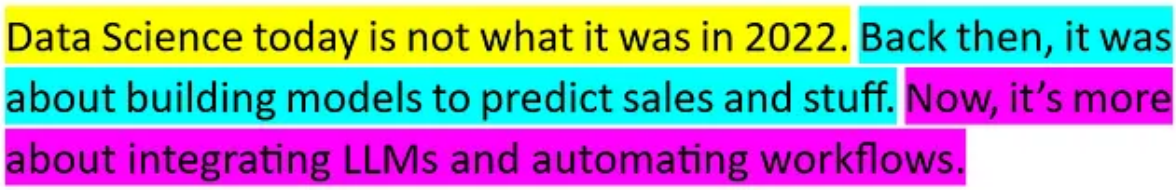
Example of sliding window chunking (Compare with fixed window chunking) | Image by author

When to use it:

- For content where ideas carry across long sentences, i.e. essays, narrative reports, free-form writings
- Just like fixed window chunking, it works for text with no structure (no heading, punctuation, schema, etc). Just be mindful of the tradeoff between token usage and broken context.

4. Sentence-based chunking

You break the text at the end of each sentence, usually marked by a full stop, question mark, or exclamation mark.



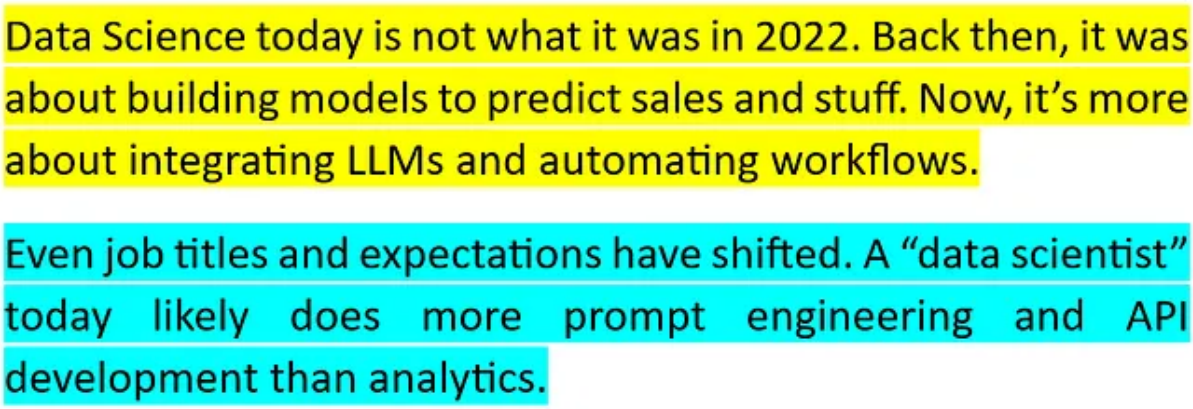
Example of sentence-based chunking | Image by author

When to use it:

- For clean, well-written text where each sentence holds a whole idea, like blogs, summaries, or documentation.
- As an initial step, to get small, focused chunks that can easily be reranked or recombined later using more complex chunking techniques.

5. Paragraph-based chunking

You split the text by paragraph (usually where there's a double line break) so each chunk holds a full idea or thought block.



Example of paragraph-based chunking | Image by author

When to use it:

- When sentence chunking feels too narrow, and you want more context per chunk.
- For documents that are already well-structured into paragraphs, like essays, blog posts, or reports

6. Page-based chunking

You treat each page as one chunk.

SOURCE: https://medium.com/@dataprincess

Example of page-based chunking | Image by author

When to use it:

- When you're working with paginated documents like scanned PDFs, slide decks, or books
- For workflows that rely on page layout i.e. need to reference page numbers in retrieval.

7. Structured chunking

You split the text based on a known structure like log entries, schema fields, HTML tags, or markdown elements.

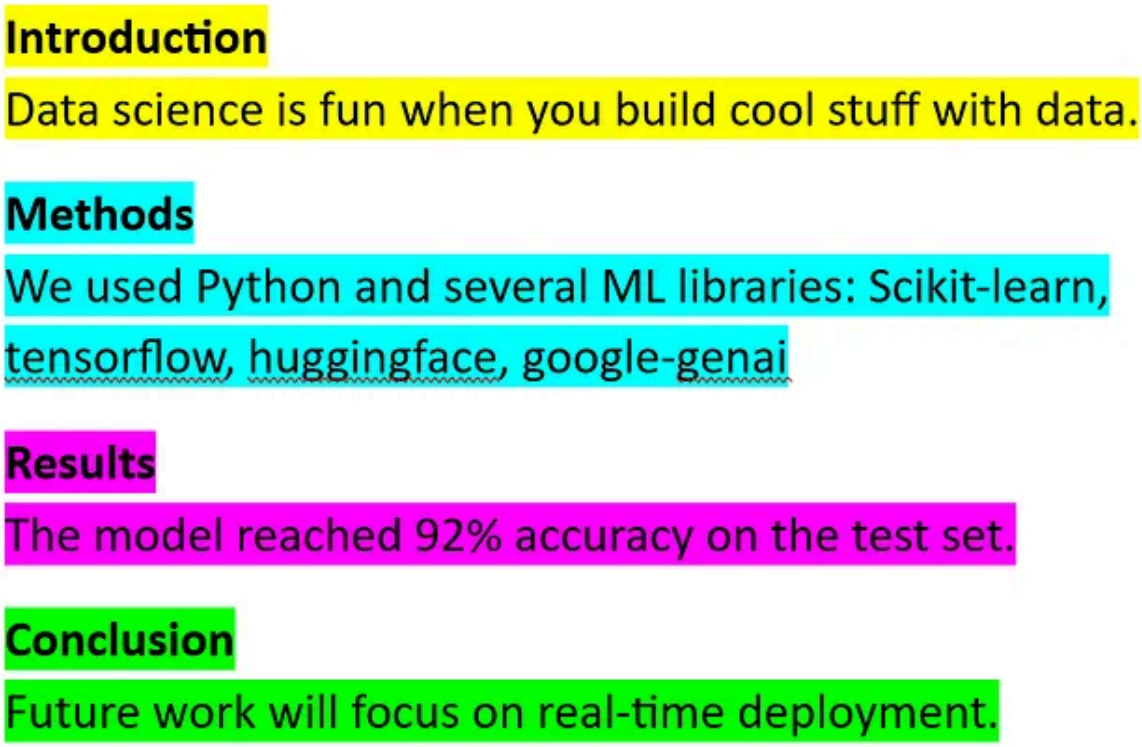
Example of structured chunking | Image by author

When to use it:

- If you're working with structured or semi-structured data like logs, JSON records, CSV, or HTML docs.

8. Document-Based Chunking:

You chunk by using the natural structure of the document (headings, subheadings, and sections).



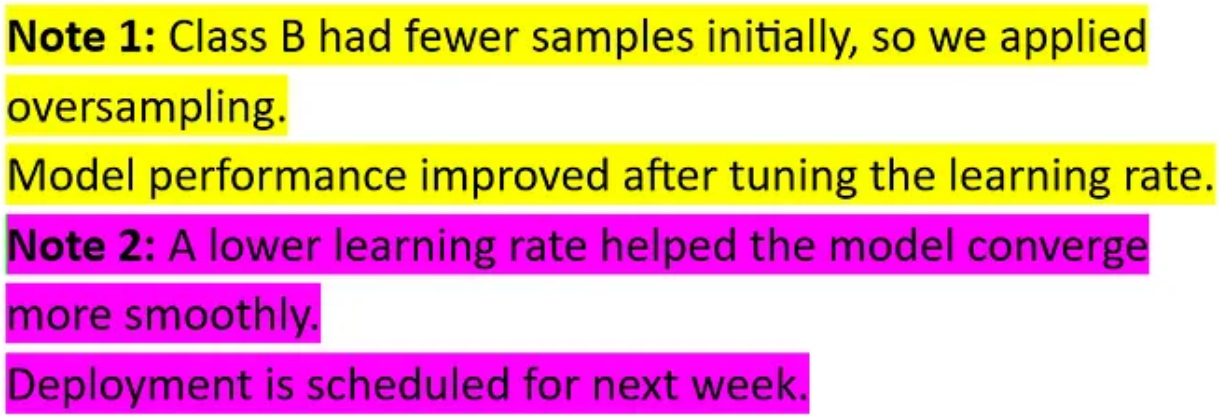
Example of document-based chunking | Image by author

When to use it:

- When your source has clear sections and headings, such as in articles, manuals, textbooks, or research papers
- As an initial step for more advanced chunking strategies, like hierarchical chunking.

9. Keyword-based chunking

You break the text wherever specific keyword(s) appear. You determine the keyword(s) beforehand and treat them as logical split points.



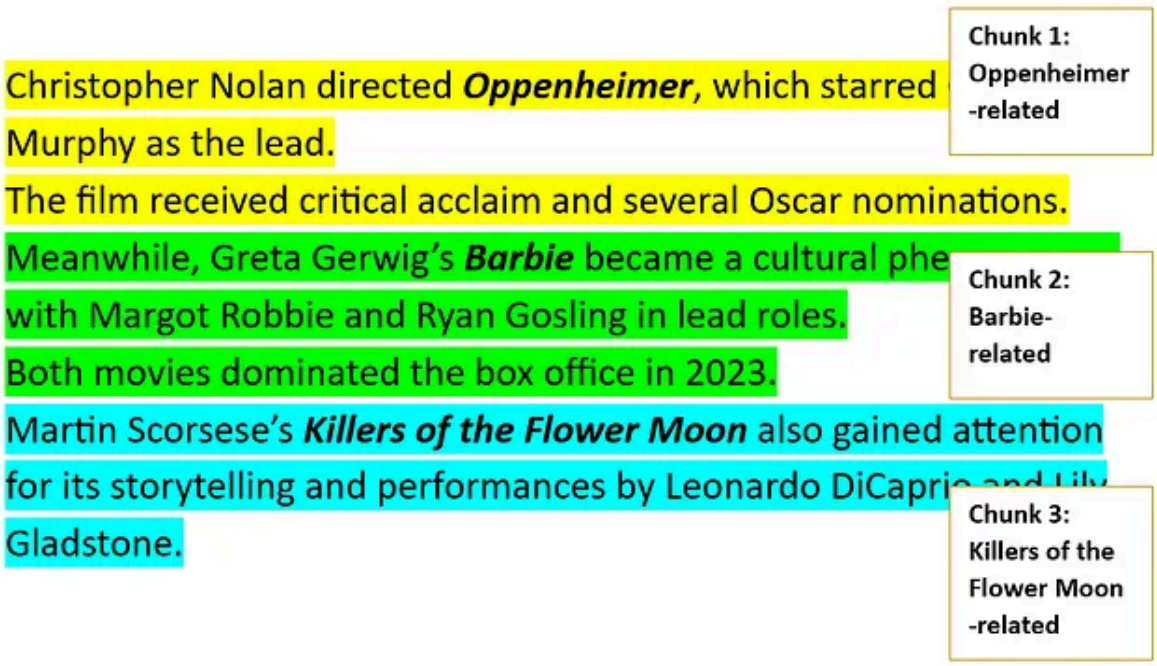
Example of keyword-based chunking (keyword is 'Note')| Image by author

When to use it:

- When heading-level splits aren't available but keyword phrases (that you know of) consistently mark new topics

10. Entity-based chunking

You use a named entity recognition (NER) model to detect entities like people, places, or products, then group related text around each one into chunks.



Example of entity-based chunking (keyword is 'Note')| Image by author

When to use it:

- For documents where specific entities matter, like news articles, legal contracts, case studies, or movie scripts.

11. Token-based chunking

You split the text by token count using a tokeniser.

You typically want to combine this technique with maybe sentence chunking to avoid splitting sentences in a way that kills context.

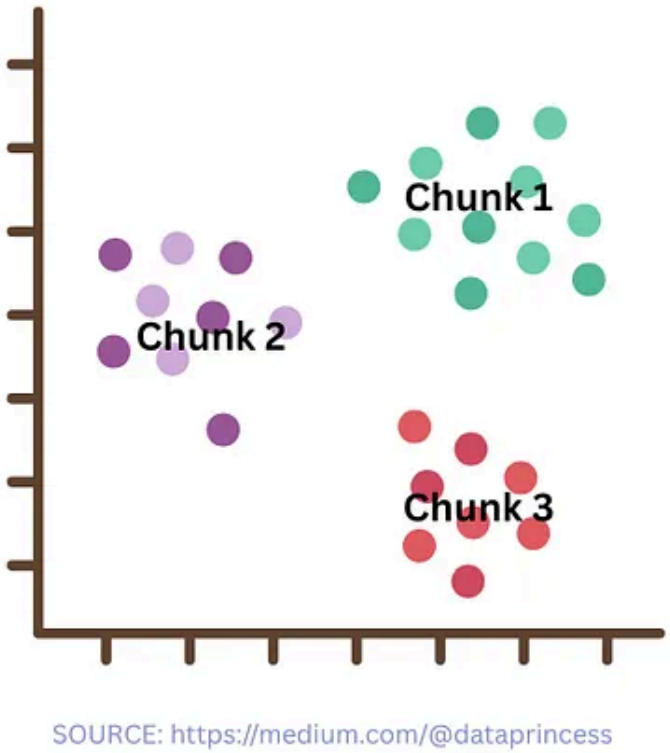
When to use it:

- For unstructured documents with no headings or paragraphing.
- When working with LLMs that have low token limits (to avoid truncation in response or processing).

12. Topic-based chunking

You break the text when the topic changes by:

- First, splitting it into smaller parts (sentences or paragraphs).
- Then, grouping related parts into a single chunk using topic modelling or clustering.



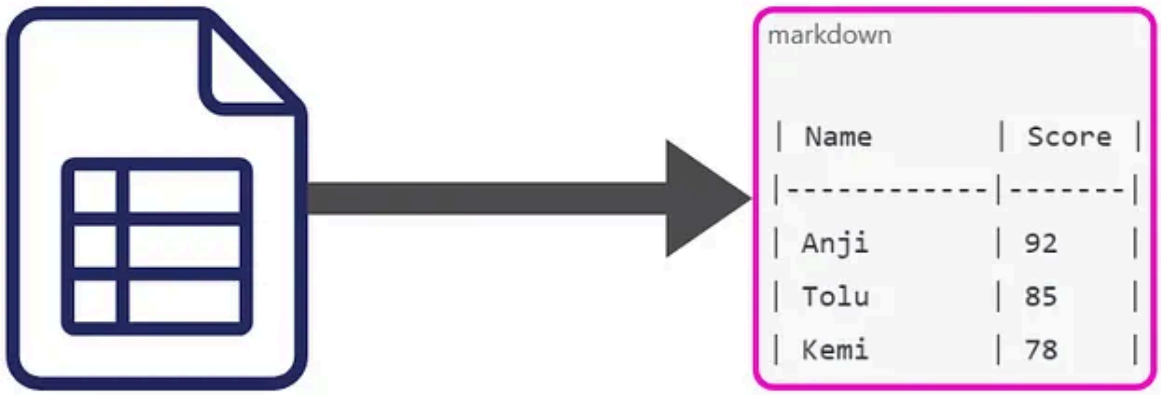
Example of topic-based chunking by Clustering | Image by author

When to use it:

- When your document covers multiple topics and you want each chunk to stay focused on one idea
- For text where the topic shifts gradually but isn't marked by explicit headings or keywords.

13. Table-aware chunking

You identify and chunk tables separately in JSON or Markdown format. It can be row by row, column by column, or the entire table.



SOURCE: https://medium.com/@dataprincess

Example of table-aware chunking | Image by author

When to use it:

- For documents that contain tables

14. Content-aware chunking

You adjust how you chunk based on the type of content, like using appropriate rules for paragraphs, tables, lists, etc.

When to use it:

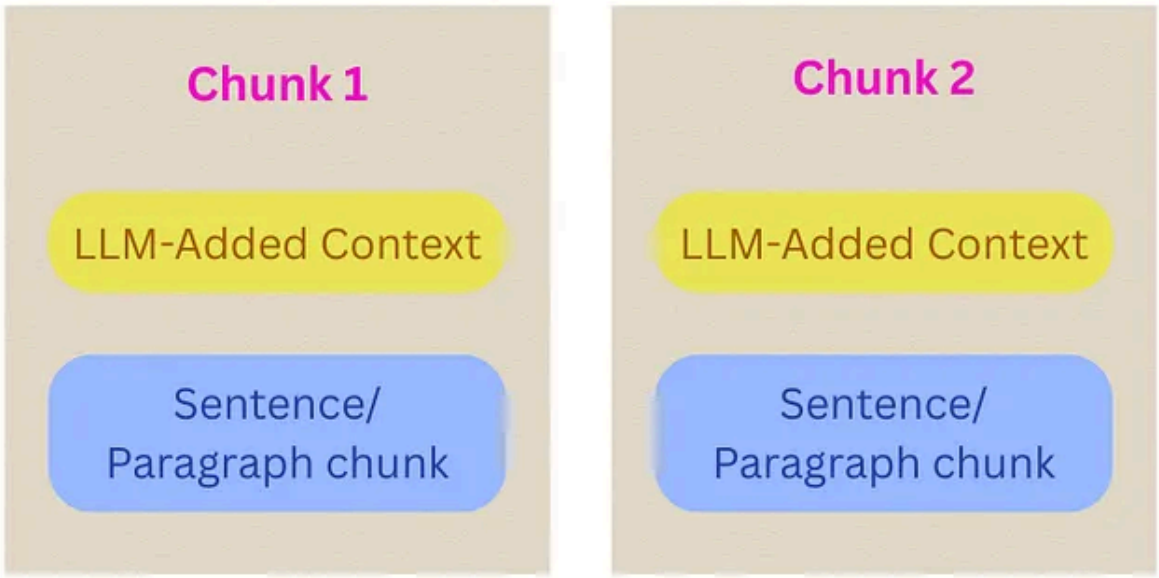
- For mixed-format documents (that contain different text structures)
- When you want chunks that respect the document's format and meaning, so tables stay whole, paragraphs stay intact, etc.

15. Contextual chunking

Use an LLM to:

- Analyse your knowledge base in part or whole

- And then add short, relevant context to each chunk before embedding.



Example of contextual chunking | Image by author

When to use it:

- When your knowledge base, in part/ whole, fits within the LLM token limit.
- For complex documents, such as financial reports and contracts.

16. Semantic chunking

You group sentences or paragraphs that talk about the same thing, using embedding similarity to keep chunks topically focused.

When to use it:

- When simpler techniques like paragraph or fixed window chunking fail
- For long documents with mixed topics

17. Recursive chunking

You begin by splitting the text at a large separator (e.g., paragraphs).

If any resulting chunk exceeds your preset chunk size limit, you recursively split those chunks further using smaller separators (e.g., sentences or words) until all chunks are within the desired size.

When to use it:

- For text with uneven or unpredictable sentence lengths, like interviews, speeches, or free-form writing

18. Embedding chunking

Usually, you chunk then embed, but here, you embed all the sentences first, then go through them in order, grouping each one with the next if their similarity is high, and splitting only when it drops below a set threshold.

When to use it:

- When your document has no structure (sentence, heading, sections, markers, etc.)
- When simpler techniques (e.g., sliding window chunking) don't cut it.

19. Agentic / LLM-based chunking

You ask the LLM to decide how to chunk the text and give it full control to break things up however it sees fit.

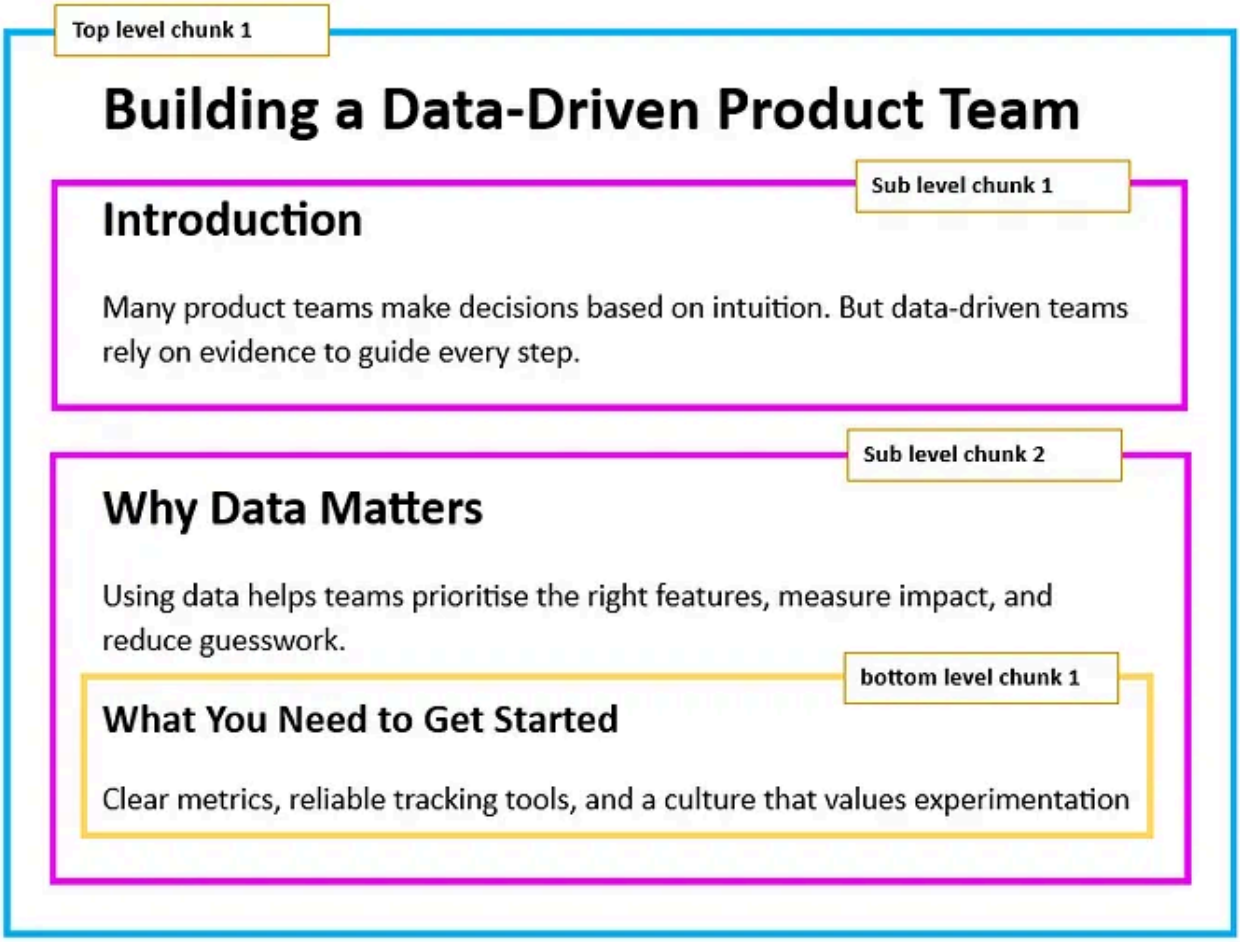
When to use it:

- When your content is complex or unstructured and needs human-like judgment to find good chunk boundaries

Note: This method can be costly or resource-intensive><

20. Hierarchical chunking

You break the text into chunks at multiple levels, such as sections, subsections, and paragraphs, so that users can retrieve info at different levels of detail.

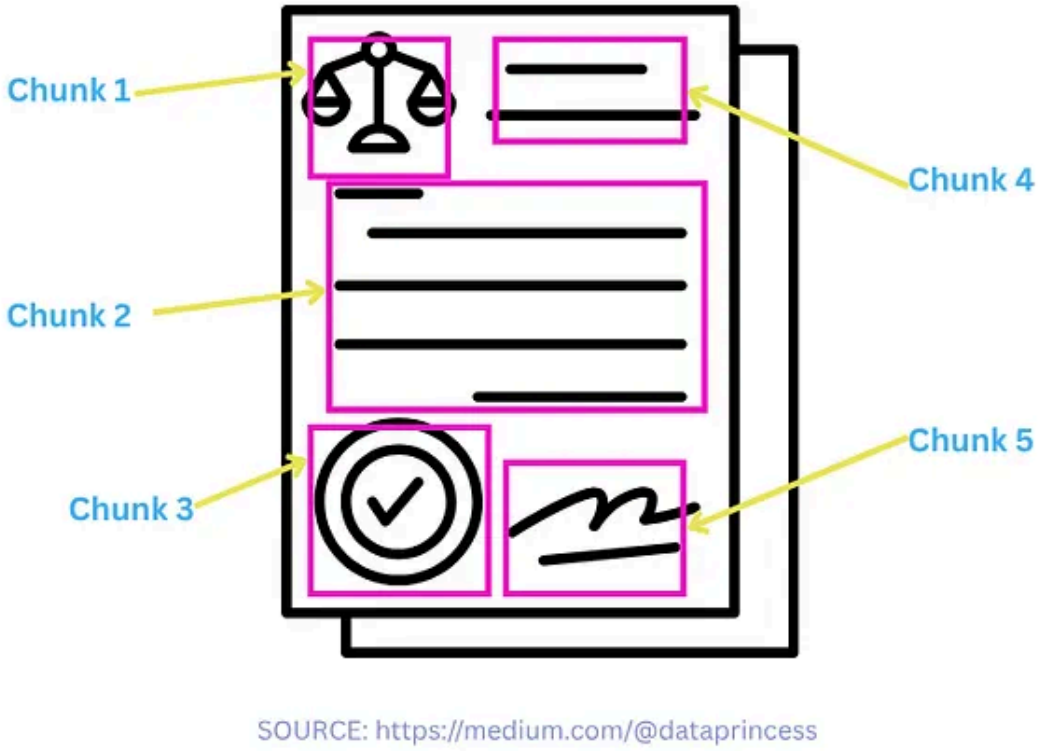


When to use it:

- For documents that have clear sections and headings, such as in articles, manuals, textbooks, or research papers.
- If you want users to explore both broad overviews and detailed info without losing context.

21. Modality-Aware Chunking

You separate different types of content (text, images, tables) in their own ways.



Example of modality-aware chunking | Image by author

BONUS: Hybrid chunking

You combine different methods, heuristics, embeddings, and/or LLMs to get more reliable chunks.

When to use it:

- When no single chunking method fits your data perfectly, so you mix approaches for better results

CTA

I hope this helps you take your RAG project to the next level.

If you enjoyed this read, give it up to **50 claps** to show some love.

Putting this together took weeks of research, writing, experimentation, and surprisingly design.

If you appreciate my work, you can tip me by clicking on the button below :D

Thank you for indulging. Bye for now.

Retrieval Augmented Gen | Artificial Intelligence | Llm | AI

Follow

Published in AI Advances

44K followers · Last published 3 hours ago

Democratizing access to artificial intelligence

Follow

Written by Anjolaoluwa Ajayi

2.4K followers · 79 following

Remote Data Scientist. I'm a big data fiend (no pun intended ><). I mostly write about Data Science, ML, and Gen AI. Might write a book soon ;)

Responses (13)

Satish

What are your thoughts?

Ciphernutz
Jul 18 (edited)

Super helpful roundup of chunking strategies! Saved this for next RAG experiment

14 1 reply Reply

Roberto Carlos
Jul 22 (edited)

Thanks for your article , it is great. Only I have a question, if you have thousands of documents, how do you choose the Best chunking function? I mean, doing by hand it is imposible.

48 1 reply Reply

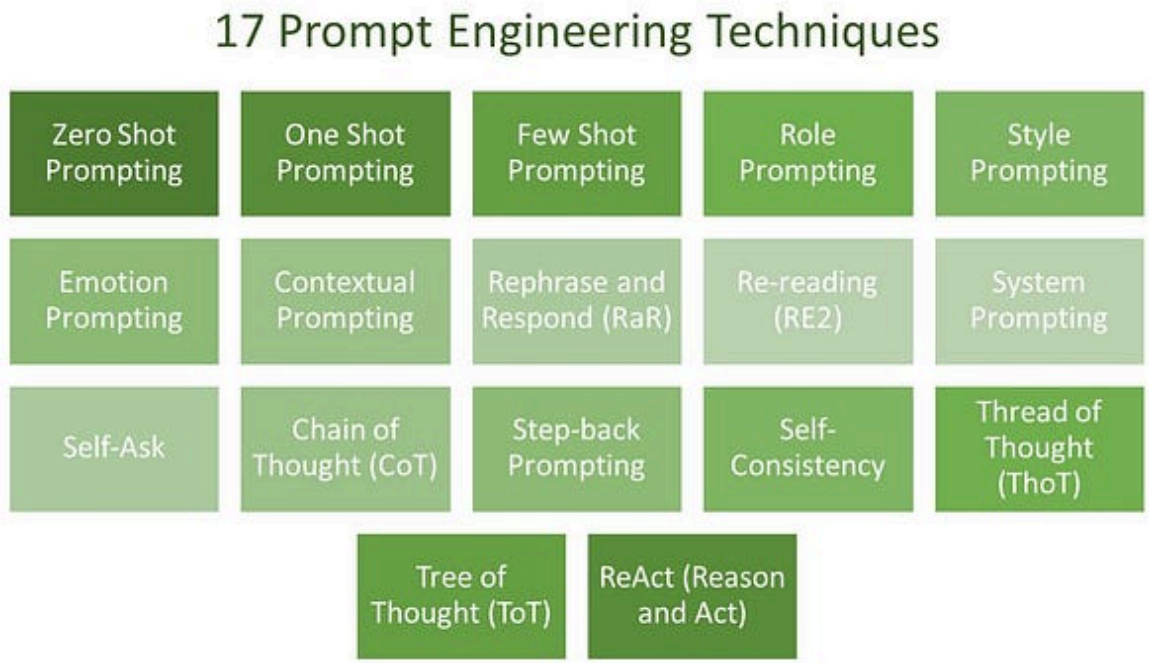
The Data Jockey
Jun 30 (edited)

As someone who is working on building GenAI-powered data governance tools, I often struggle with choosing the right chunking approach to balance context retention and latency. Your explanation has given a much clearer decision. I'll be applying these insights in my upcoming LangChain + Azure GPT workflow!

14 1 reply Reply

See all responses

More from Anjolaoluwa Ajayi and AI Advances



In AI Advances by Anjolaoluwa Ajayi

17 Prompt Engineering Techniques and When to Use Them

From Easy to Advanced with Examples

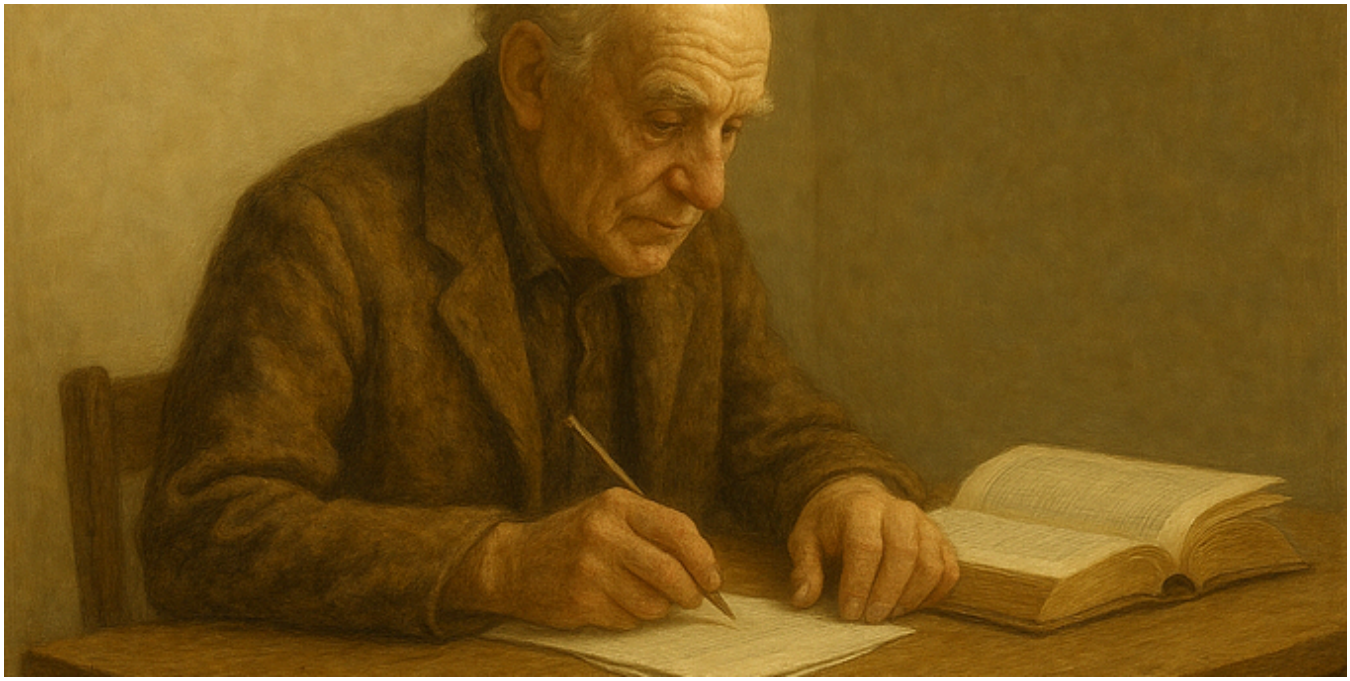
Jan 26 1.4K 12

In AI Advances by Kenny Vaneetvelde

The Hidden Costs of LangChain, CrewAI, PydanticAI and Others: Why Popular AI Frameworks Are Failing...

After 15 years in software development and countless hours wrestling with AI frameworks, I had reached a breaking point. The promise of...

Jul 13 689 26

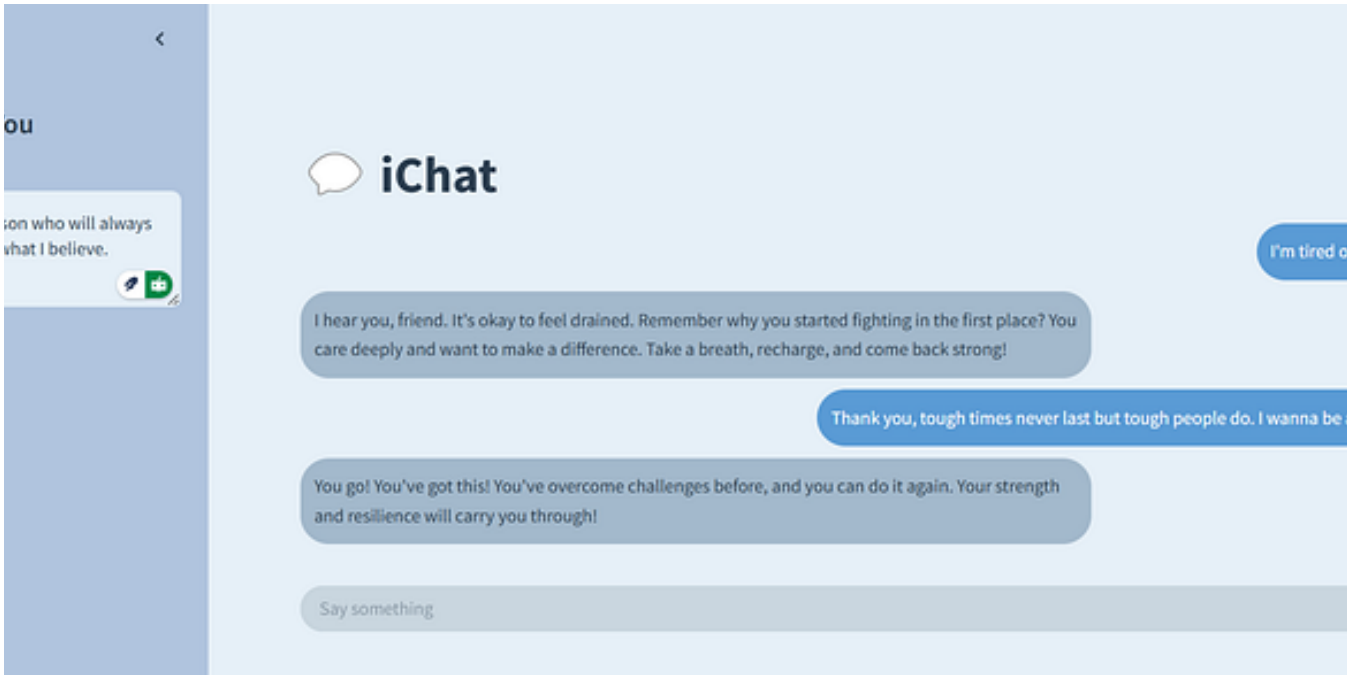


In AI Advances by Paul Siemers, PhD

AI Breaks the Chinese Room

LLMs render Searle's famous thought experiment obsolete

Jul 22 1.4K 84



In Level Up Coding by Anjolaoluwa Ajayi

Your Streamlit Apps Don't Have to Suck

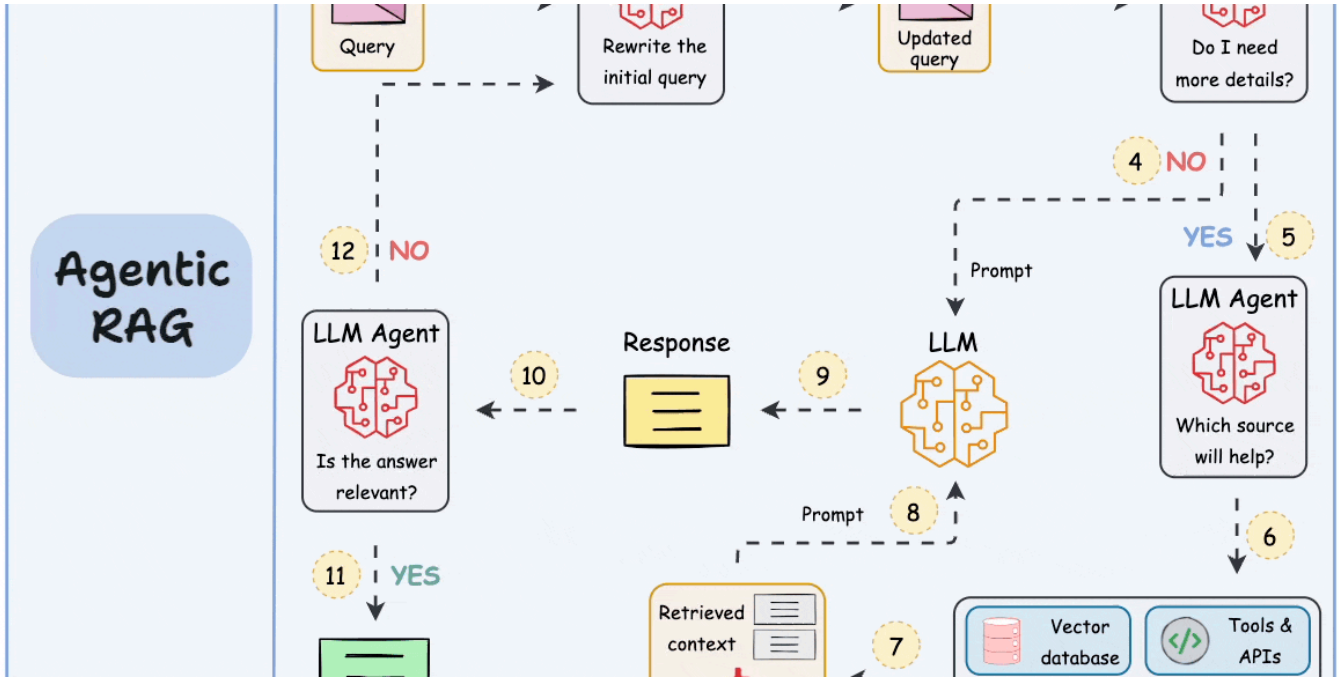
6 Simple Ways to Customize Your Streamlit Apps

Feb 14 536 1

See all from Anjolaoluwa Ajayi

See all from AI Advances

Recommended from Medium

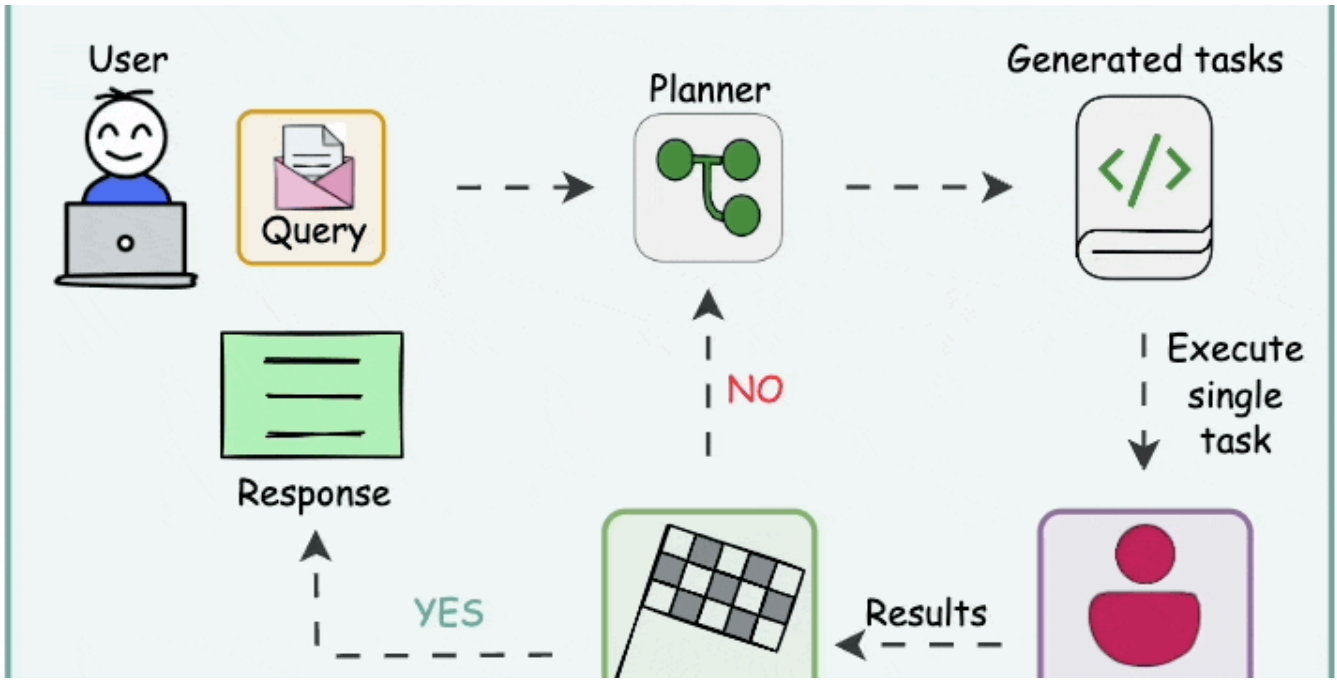


In Artificial Intelligence in Plain English by Piyush Agnihotri

Building Agentic RAG with LangGraph: Mastering Adaptive RAG for Production

Build intelligent RAG systems that know when to retrieve documents, search the web, or generate responses directly

Jul 20 1.4K 18

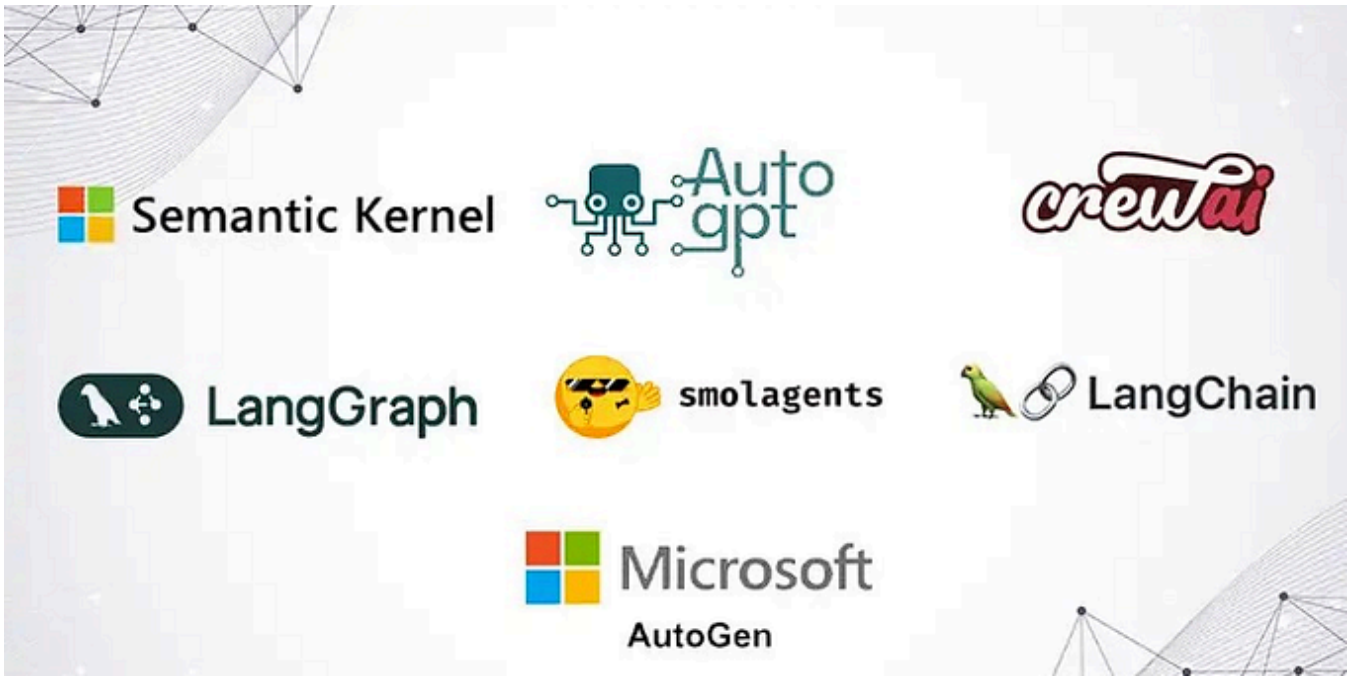


In Data Science Collective by Paolo Perrone

Stop Prompting, Start Designing: 5 Agentic AI Patterns That Actually Work

When I first started working with LLMs, I thought it was all about writing the perfect prompt. Feed it enough context and—boom—it...

Jul 23 1.2K 15



In AIGuys by Vishal Rajput

Leave Agentic AI Frameworks And Build Agents From Scratch

I'll be honest with you, I hate most agent-based AI workflows; they are simply unusable in the real world at scale. Despite the...

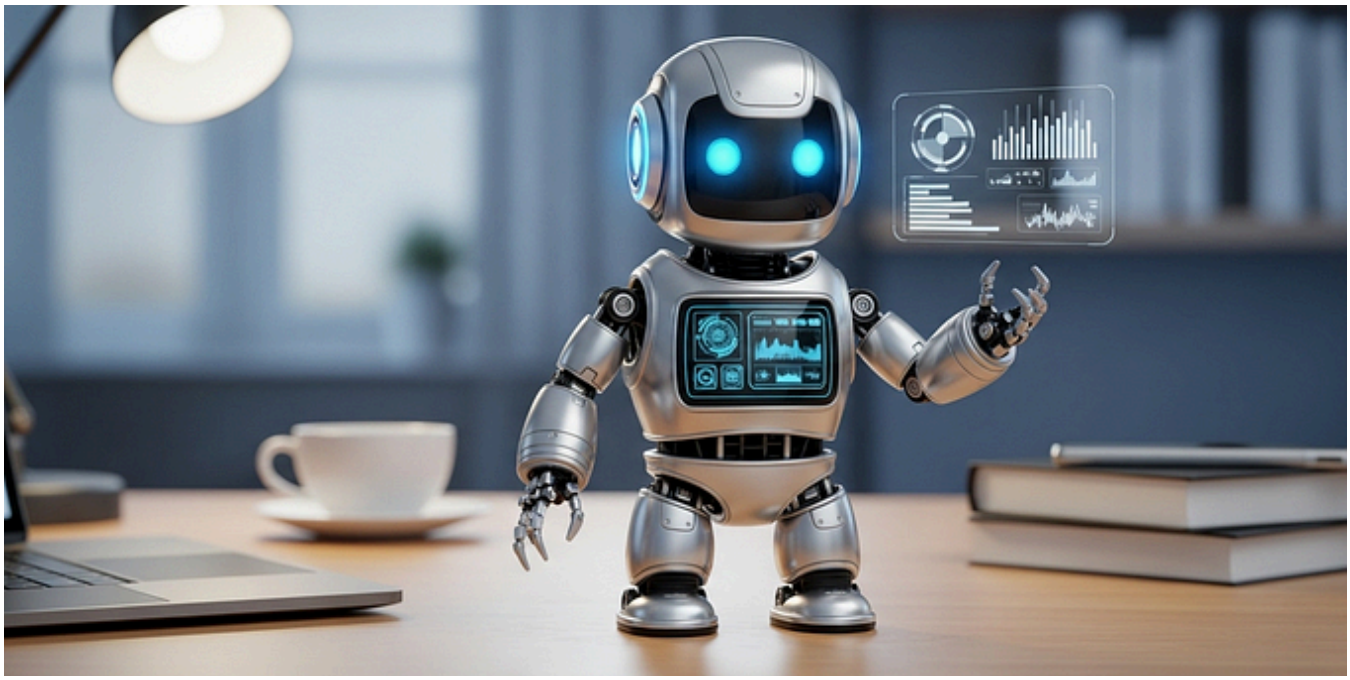
Jul 28 408 14

In JavaScript in Plain English by Hassan Trabelsi

I Built an AI Team That Codes Itself, And You Can Too

No prompts. No babysitting. Just real coding agents, building full-stack apps while you sleep.

Jul 27 1.2K 33



Harisudhan.S

AI Agent vs Agentic AI: Understand The Actual Difference

We hear a lot about AI agents which can book meetings using tools, search the internet, even generate code. And then came an another term...

Jul 6 501 36

Sohail Saifi

The Death of Agile: Why Tech Giants Are Abandoning Scrum and What They Use Instead

Do you recall when Scrum was the rage? When was it mandatory for all businesses to have certified Agile Coaches and Scrum Masters? When...

Jul 20 4.5K 255

See more recommendations