

Agora — RAG System for AI Governance Documents

Ask a question about global AI governance and get an answer with sources, not a guess.

BUILT WITH

RAG

Google Gemini

Pinecone

FastAPI

Streamlit

Upstash Redis

Python

1-5

sub-queries per question

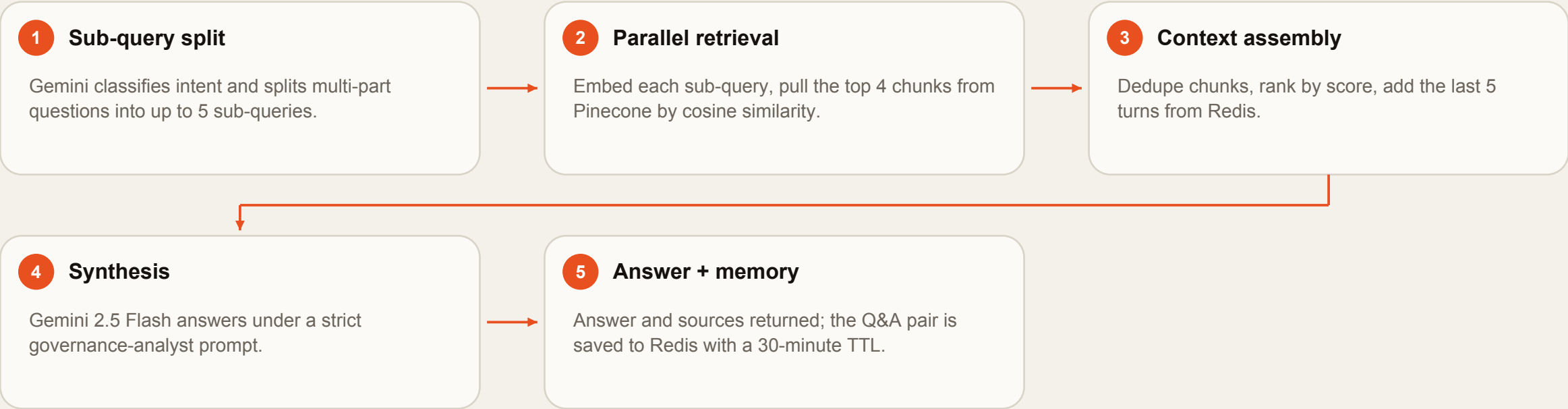
4

chunks retrieved per sub-query

30 min / 7 days

model memory / UI history

How it works



WHAT EACH PART DOES

API layer (FastAPI)	Checks each request and hands it to the core. No business logic lives here.
RAG core	Reads documents, splits them into chunks, embeds them and searches Pinecone.
Chat engine	Splits a question into sub-queries and writes the final answer.
Memory (Redis)	Keeps recent turns so follow-up questions make sense.
Prompt file	Every prompt template in one YAML file, loaded once at startup.



Key decisions

01

Split the question before searching

One search over a multi-part question averages the parts and quietly drops one half of the answer. Splitting into up to 5 sub-queries and searching in parallel fixes that.

TRADE-OFF

Adds about 2-3 seconds. For a governance tool, completeness beats speed.

02

Chunk size tuned to the documents

The default 1,200-token chunk turned each short document into a single chunk, which destroyed search precision. 500 tokens with 20% overlap gave 36 distinct chunks from 10 files.

03

Two separate memories

A 30-minute, 5-message window feeds the model, because older context dilutes retrieval. A separate 7-day store backs the UI so users can reopen old sessions.

TRADE-OFF

Very old sessions lose model context even though the history stays visible.

04

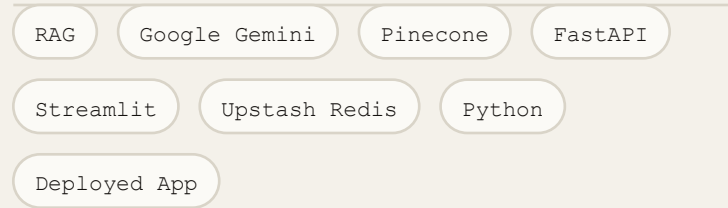
Failures are loud, not silent

An early version returned empty vectors when embedding failed, which Pinecone rejected with an unclear error. It now retries three times with backoff, then fails visibly.



Stack, status and what is next

BUILT WITH



KNOWN LIMITS

- Scanned or image-only PDFs cannot be read.
- Separating document sets by namespace is trust-based, not cryptographic, so it is unsuitable for sensitive multi-tenant data.

NEXT STEPS

- Hybrid search (keyword plus semantic) to catch exact statute references.
- Cross-encoder reranking on the top 20 results before picking the final 4.
- Metadata filters such as jurisdiction, document type and year.

LIVE DEMO

<https://agora-rag.streamlit.app/>

SOURCE

<https://github.com/wailethewave/Agora-RAG>