

OpenAI

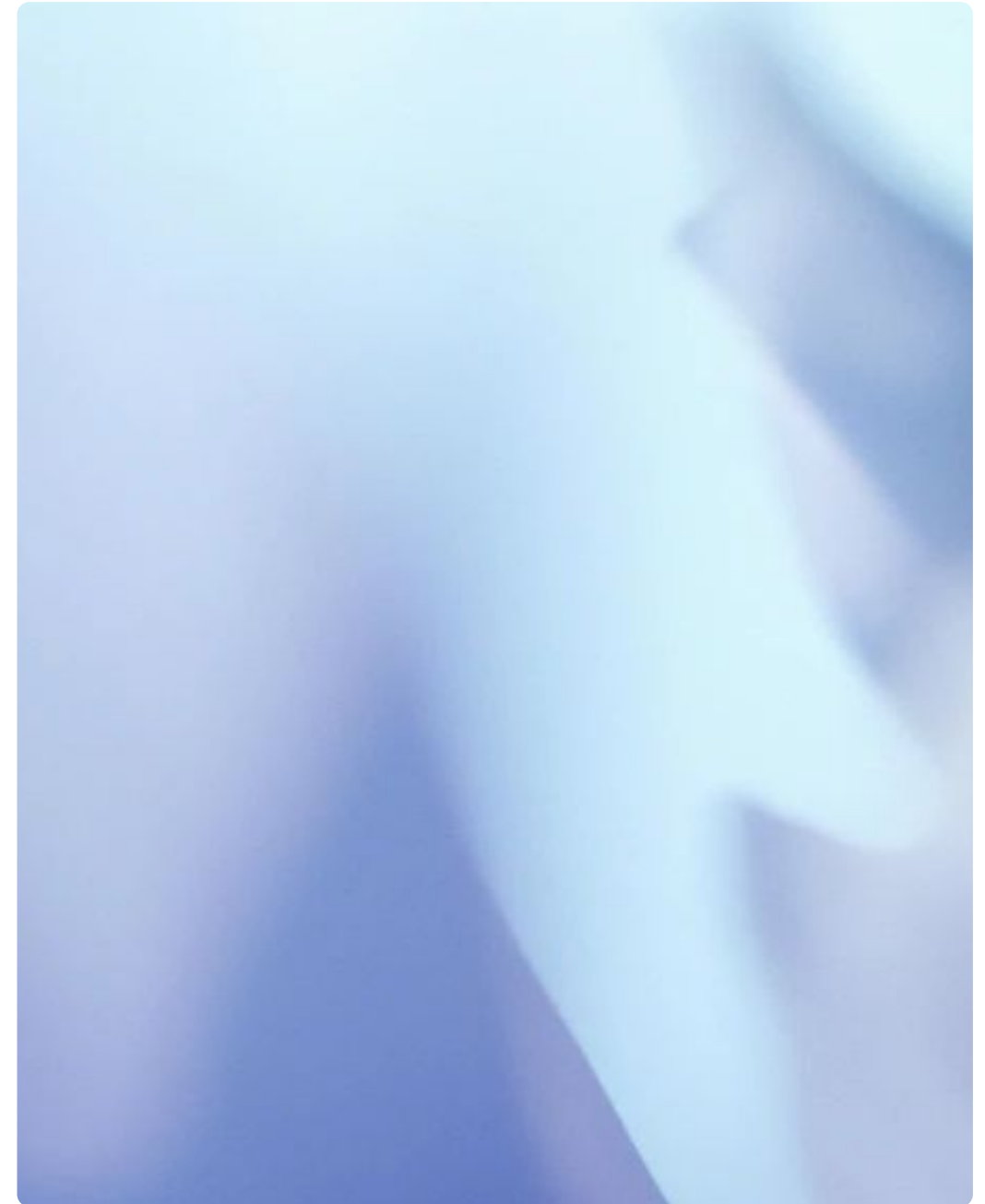
# Retrieval-Augmented Generation (RAG)

Builder Training



Palash Matey

Builder AI Deployment Manager

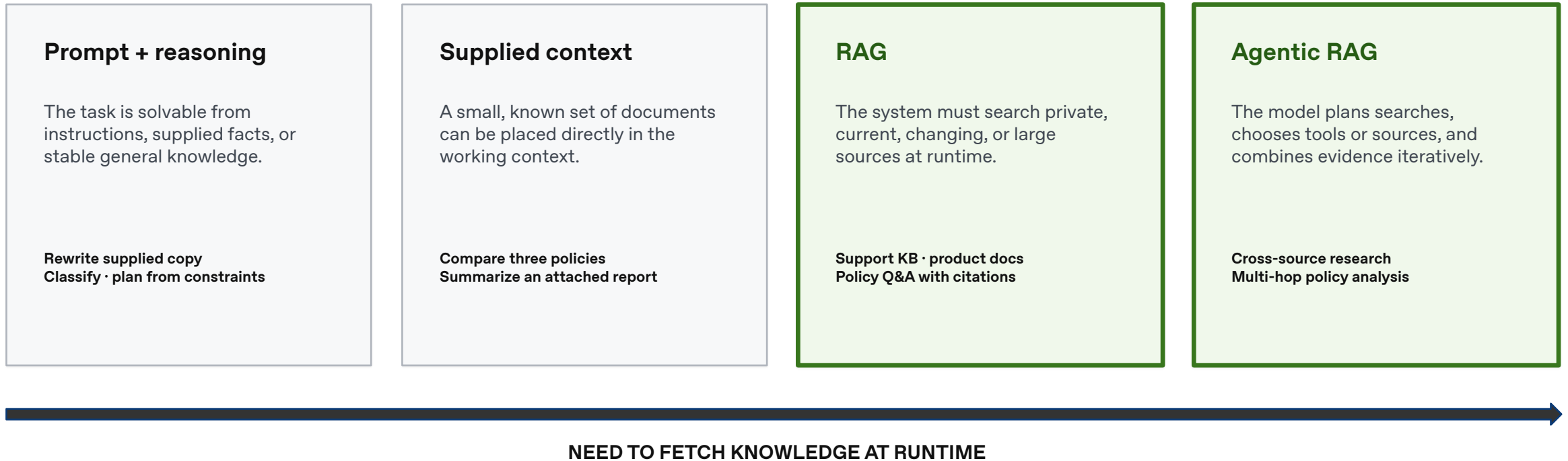


- 01 RAG overview
- 02 Technical patterns
- 03 Evaluation
- 04 Q&A
- 05 Resources

# When you need RAG

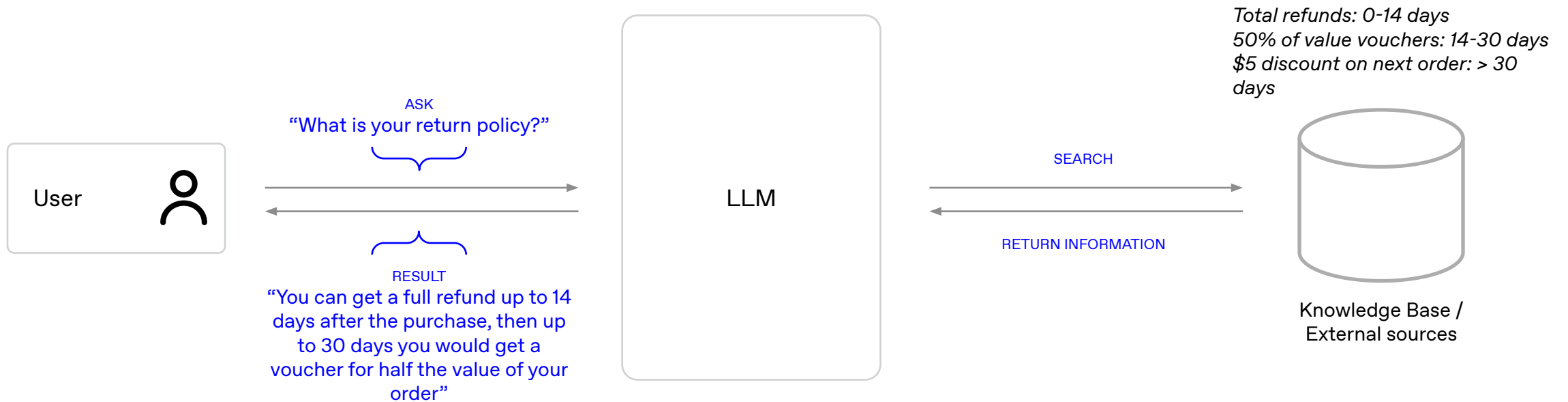
Use retrieval when the answer requires knowledge the model does not reliably have in context

## KNOWLEDGE REQUIREMENT SPECTRUM



# What is RAG?

Retrieve information to Augment the model's knowledge and Generate the output



# Technical patterns



## Data preparation

- Chunking
- Embeddings
- Augmenting content



## Input processing

- Input augmentation
- NER
- Embeddings



## Retrieval

- Search
- Multi-step retrieval
- Re-ranking



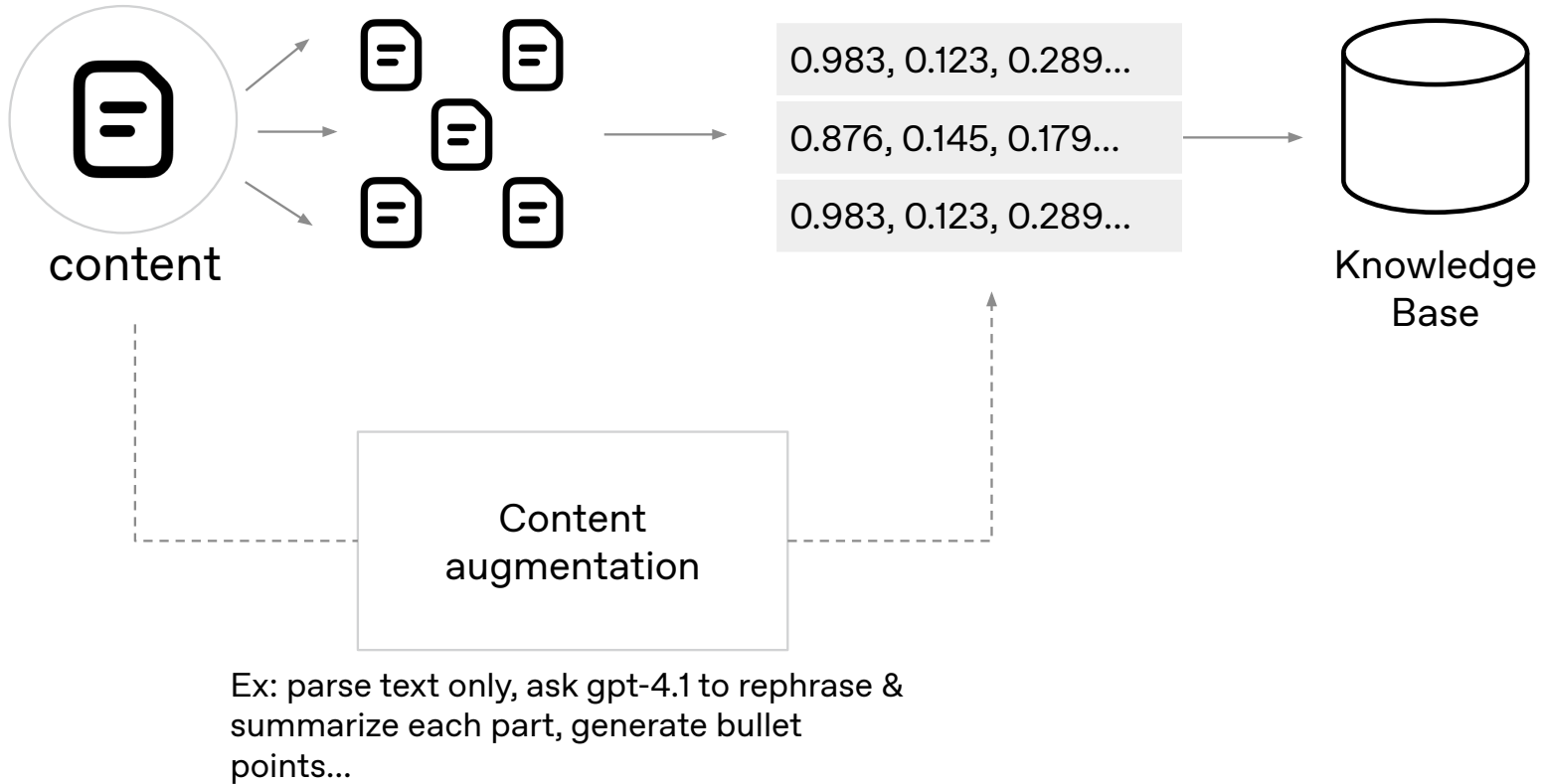
## Answer Generation

- Context window
- Optimization
- Safety checks

# Technical patterns

## Data preparation

chunk documents into multiple pieces for easier consumption



### Best practices

Index only approved, current sources with clear ownership.

Attach scope, date, permissions, and source link to every chunk.

### Common pitfalls

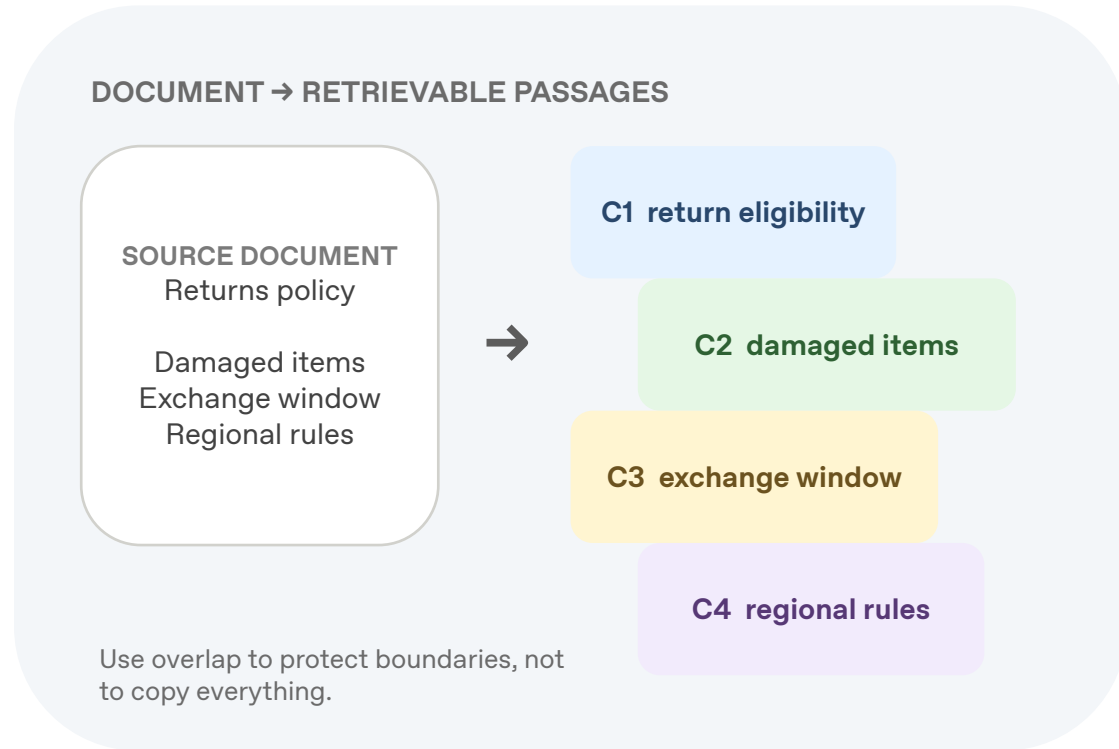
Duplicate or conflicting sources produce inconsistent answers.

Retired content and outdated permissions remain retrievable.

# Technical patterns

## Data preparation: Chunking

Break long documents into passages that retrieval can surface precisely while optimizing for cost and latency.



### CONSIDERATIONS

#### 01 OVERLAP

Protect boundaries without duplicating too much context.

#### 02 SIZE

Balance retrieval precision with enough local context.

#### 03 BOUNDARIES

Split on logical sections when structure helps retrieval.

# Technical patterns

## Data preparation: Chunking Strategies

1

### Fixed size + overlap

Reliable baseline for mixed or irregular content. Tune size and overlap with evals.

2

### Sentence windows

Keep complete sentences together for prose. Add overlap when ideas cross a boundary.

3

### Structure-aware

Use headings, sections, paragraphs, and tables when the document format is reliable.

4

### Semantic boundaries

Split on topic shifts when simpler boundaries mix unrelated ideas. Higher ingestion cost.

# Technical patterns

## Data preparation: Embeddings

An embedding is a high-dimensional vector. Texts with related meaning tend to point in similar directions—even when their wording differs.

### QUERY + PASSAGES

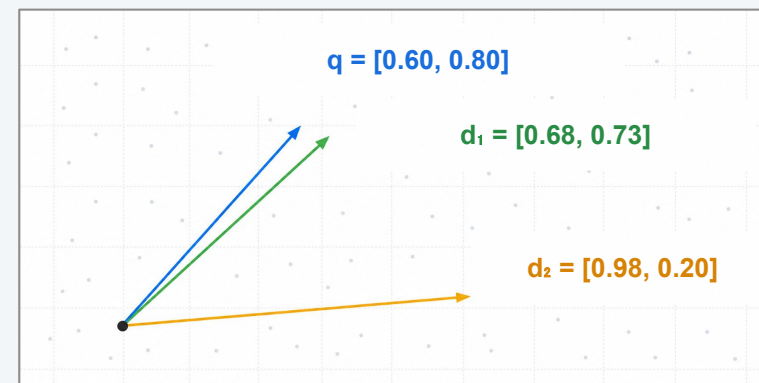
q "How do I return a damaged item?"

d<sub>1</sub> "Defective products can be sent back within 30 days."

d<sub>2</sub> "Where can I update my billing address?"

EMBEDDING MODEL

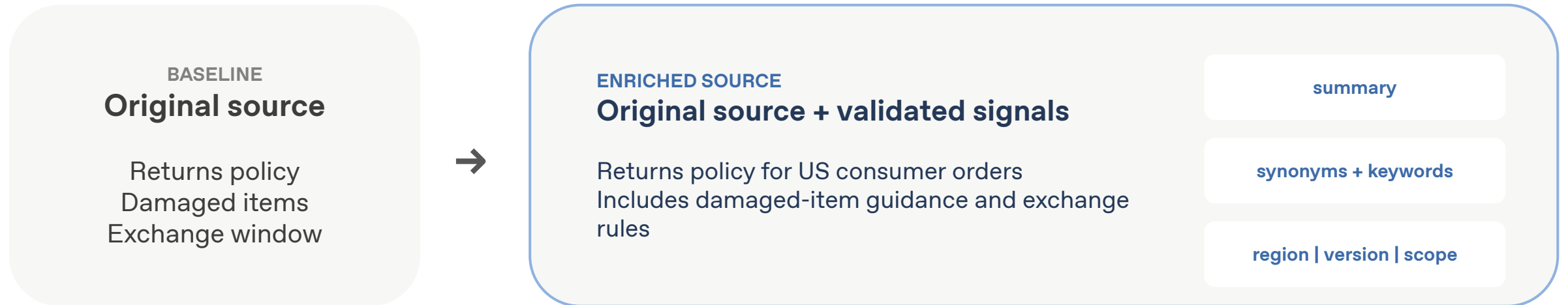
### ILLUSTRATIVE 2D VECTORS



# Technical patterns

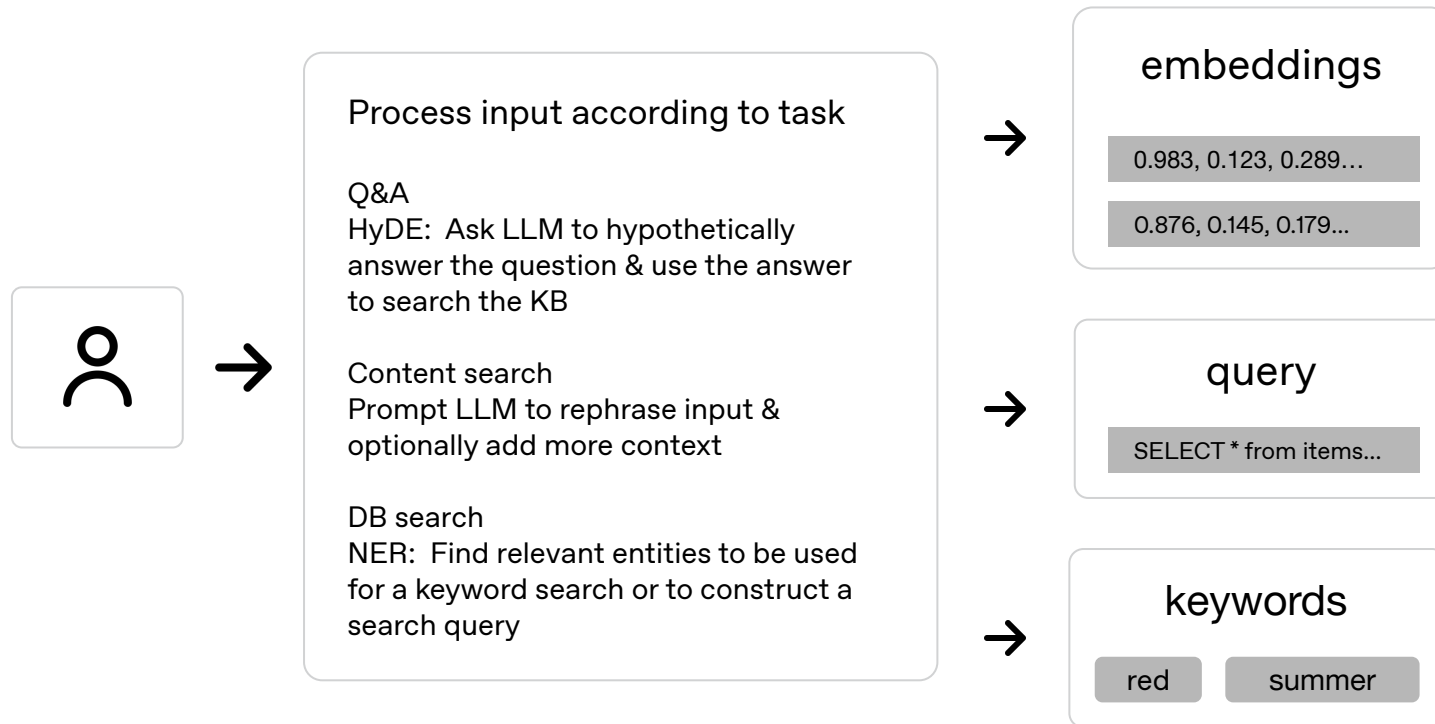
## Data preparation: Augmenting Content

Start with the original source. Add searchable signals only when retrieval tests show a gap.  
Review generated summaries, descriptions, and transcripts before indexing.



# Technical patterns

## Input processing



## Best practices

Consider how to transform the input to match content in the database

Consider using metadata to augment the user input

## Common pitfalls

Query rewrites can lose the user's intent or constraints without improving retrieval.

# Technical patterns

Input processing: Input Augmentation

Four techniques shape the user query before retrieval. Start with the simplest transformation that addresses a measured failure.

## 01 Query expansion

### REFRAME THE REQUEST

Rephrase the request using source language while preserving intent.

“My MX-200 arrived broken”  
→ “MX-200 damaged-item return policy”

## 02 Decompose

### SPLIT MULTIPART REQUESTS

Retrieve independent questions separately, then merge the evidence.

“Can I return it and who approves exceptions?”  
→ A. return eligibility  
B. approval authority

## 03 NER + filters

### EXTRACT HARD CONSTRAINTS

Extract product, region, date, version, or access fields for filtering.

“MX-200 bought in California last month”  
→ product=MX-200  
region=CA | age≤30d

## 04 HyDE

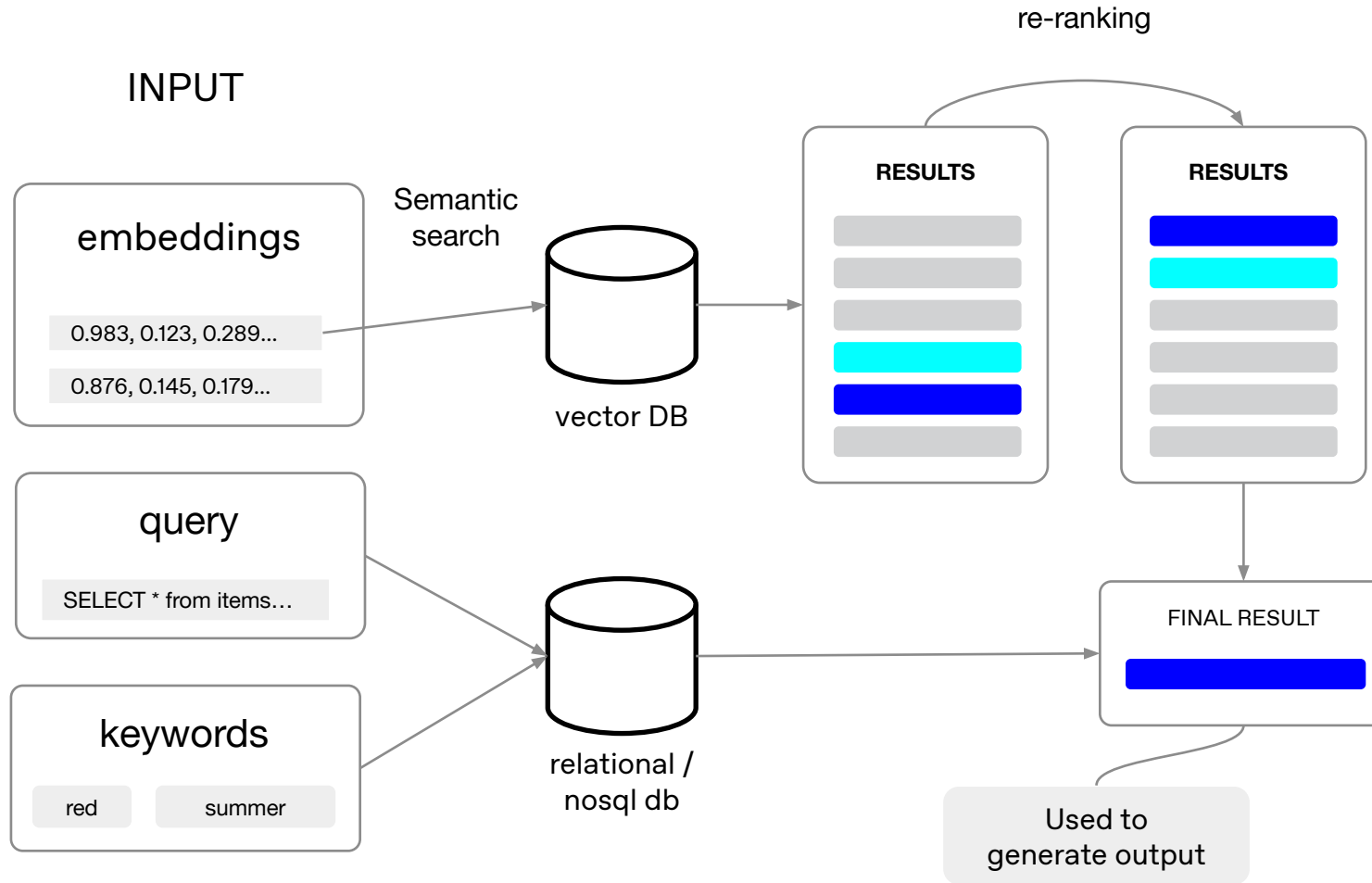
### TEST A SEARCH PROXY

Generate an answer-shaped proxy when a short conceptual query retrieves weak evidence.

“What protections cover damaged items?”  
→ Embed a hypothetical policy answer

# Technical patterns

## Retrieval



### Best practices

Use a combination of semantic search and deterministic queries where possible  
+ Cache output where possible

### Common pitfalls

The wrong elements could be compared when looking at text similarity, that is why re-ranking is important.

# Technical patterns

Retrieval: Search

## Semantic search

### Embeddings + vector similarity

Find paraphrases and conceptually similar passages.

## Keyword search

### BM25 lexical scoring

Scores documents based on exact term frequency and rarity.

## Attribute filters

### Pre-filter structured fields

Limit by region, version, audience, or access scope.

## Hybrid search

### Semantic + keyword fusion

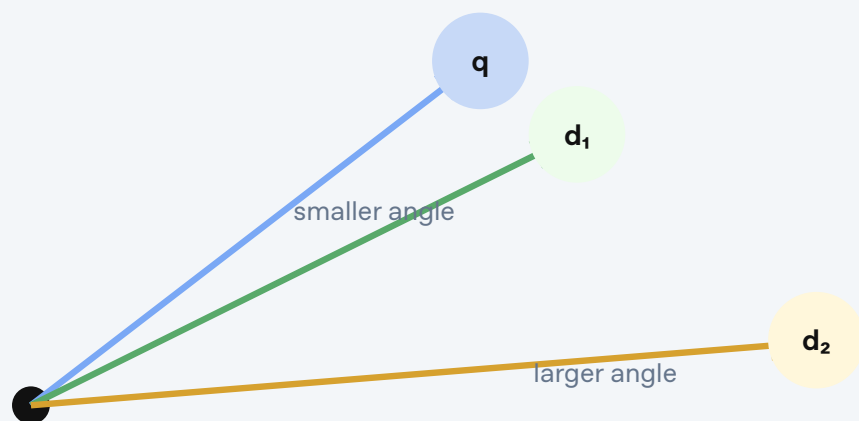
Combine signals with RRF; tune their relative weight.

# Technical patterns

Retrieval: how semantic and keyword search score relevance

Semantic compares vector direction. BM25 rewards exact terms by frequency, rarity, and document length.

## SEMANTIC SEARCH · COSINE



$$\text{sim}(q, d) = (q \cdot d) / (\|q\| \|d\|) \quad \text{dcos} = 1 - \text{sim}$$

## KEYWORD SEARCH · BM25

$$\text{BM25} \approx \text{rarity} \times \text{useful frequency} \times \text{length adjustment}$$

### RARE TERM

2 / 10,000 docs

**ERR-4297 → HIGH**

### COMMON TERM

7,800 / 10,000 docs

**error → LOW**

**Repeat matches saturate; long passages are normalized.**

**Semantic finds paraphrases. Keyword protects exact identifiers. Hybrid fuses rankings—not raw scores.**

# Technical patterns

Retrieval: Combine search signals and controls

| QUERY PATTERN                                                                 | EXAMPLE                                          | RETRIEVAL RECIPE        |
|-------------------------------------------------------------------------------|--------------------------------------------------|-------------------------|
| <b>01 NATURAL-LANGUAGE CONCEPT</b><br>Conceptual language, no exact key.      | “What protections cover delayed deliveries?”     | SEMANTIC                |
| <b>02 EXACT IDENTIFIER</b><br>Add lexical matching.                           | “What does ERR-4297 mean?”                       | SEMANTIC KEYWORD        |
| <b>03 STRUCTURED SCOPE</b><br>Apply validated metadata constraints.           | “MX-200 policy for California purchases in 2026” | SEMANTIC FILTER         |
| <b>04 MIXED, HIGH-STAKES QUERY</b><br>Combine signals; rerank only if needed. | “ERR-4297 warranty exception for MX-200 in CA”   | SEMANTIC KEYWORD FILTER |

# Q&A

# Technical patterns

## Retrieval: Multi-step retrieval

Single-step retrieval answers from one place. Use retrieval chaining when the answer must be assembled from multiple places.

“WHICH TEAMS DEPEND ON PROJECT AURORA, AND WHO MANAGES THEM?”

### 01 FIND PROJECT

Search: "Project Aurora"  
Learn: Finance Analytics +  
RevOps



### 02 FOLLOW LINKS

Search both teams  
Learn: Sarah + David



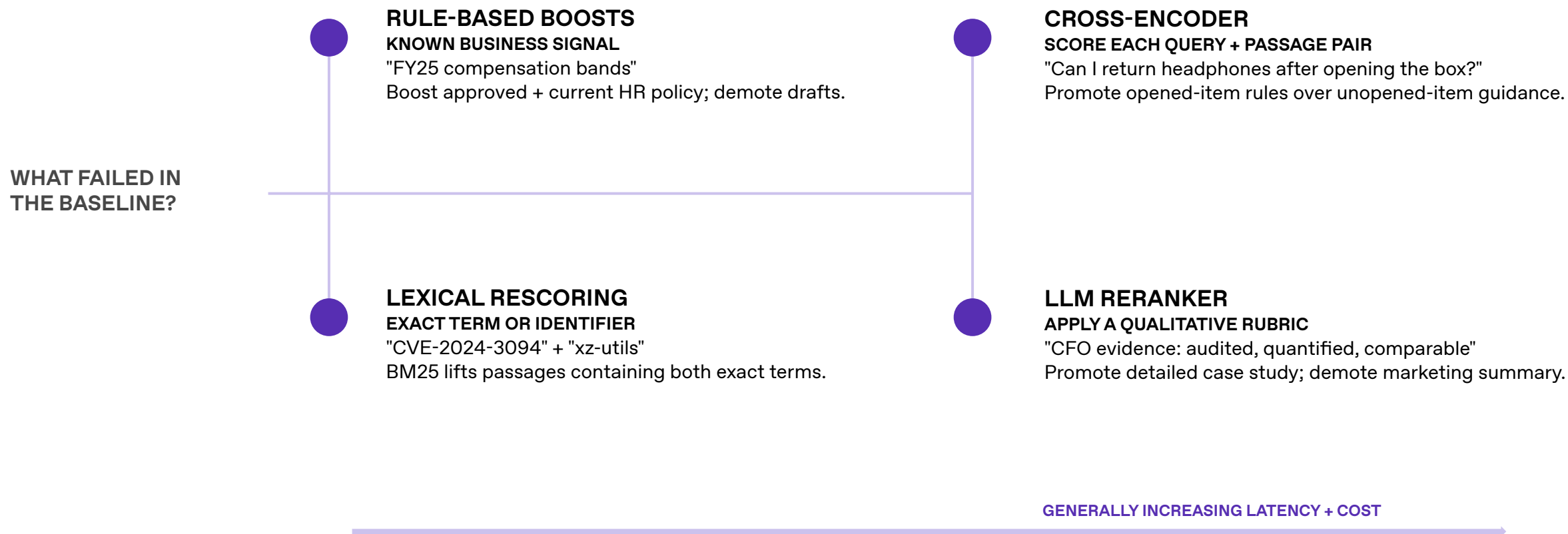
### 03 ASSEMBLE ANSWER

Sarah + David manage  
Aurora's dependent teams

# Technical patterns

## Retrieval: re-ranking

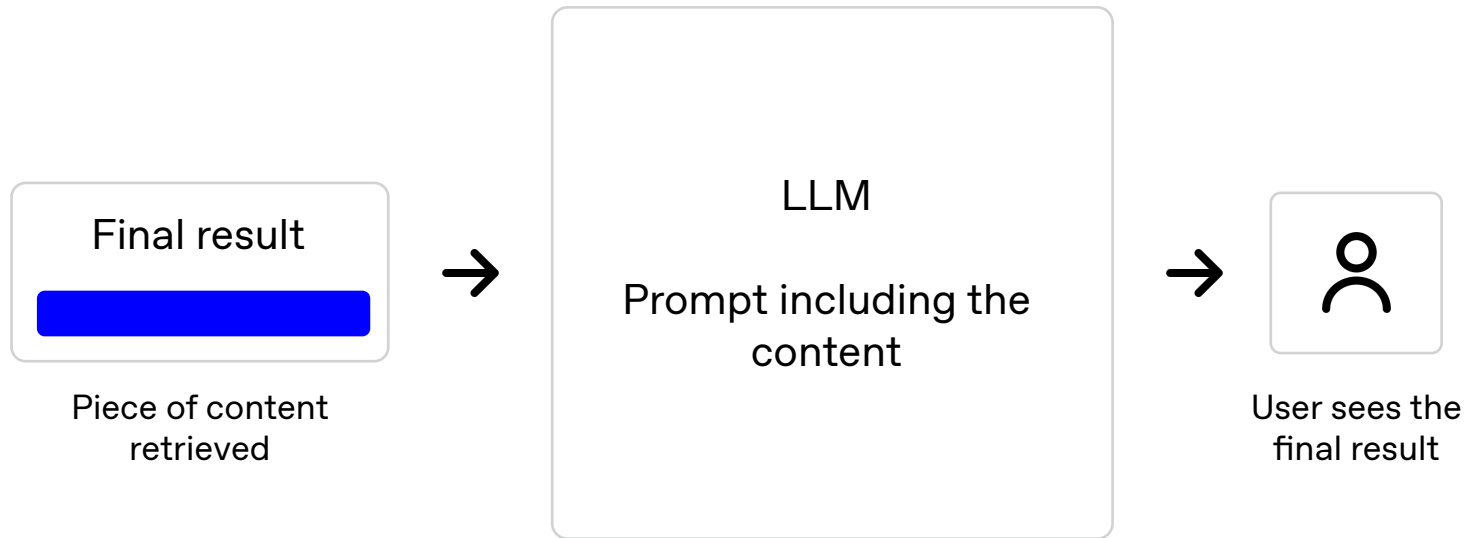
Each technique fixes a different ranking failure. The examples show why the top result changes; combine techniques when useful.



VALIDATE THE PICK | compare MRR / nDCG | p95 latency | cost per query

# Technical patterns

## Answer Generation



### Best practices

Evaluate performance after each experimentation to assess if it's worth exploring other paths

+ Implement guardrails if applicable

### Common pitfalls

Not paying attention to the way the model is prompted

# Technical patterns

Answer generation: Spend context deliberately

The model has a finite context window. Every extra token competes with the evidence and response space that matter most.

## THE CONTEXT WINDOW IS A FINITE BUDGET

**INSTRUCTIONS**  
Task + answer  
rules

**REQUEST**  
What is being  
asked

**HIGH-SIGNAL EVIDENCE**  
Only chunks that help answer

**HISTORY**  
Only what  
matters

**RESPONSE ROOM**  
Leave space to answer

### DECISION RULE

Retrieve enough evidence to answer well. Do not flood the model with marginal context.

# Technical patterns

Answer generation: optimization

Work from the simplest intervention outward. Use evaluations after each change so the team knows which experiment created the improvement and whether another step is justified.

## PROMPT ENGINEERING

Clarify the task, structure,  
and required evidence  
behavior.

**1. START HERE**

## FEW-SHOT EXAMPLES

Show difficult inputs and the  
expected output format.

**2. ADD WHEN NEEDED**

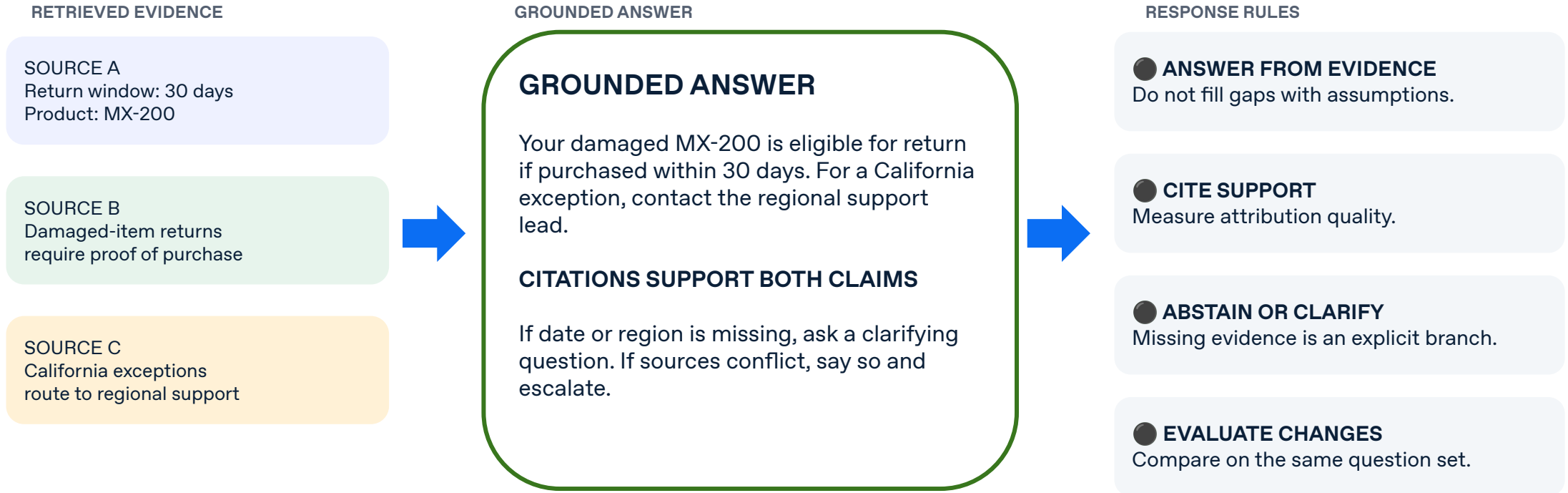
## EVALUATION LOOP

Test each experiment against  
representative questions.

**3. MEASURE EACH CHANGE**

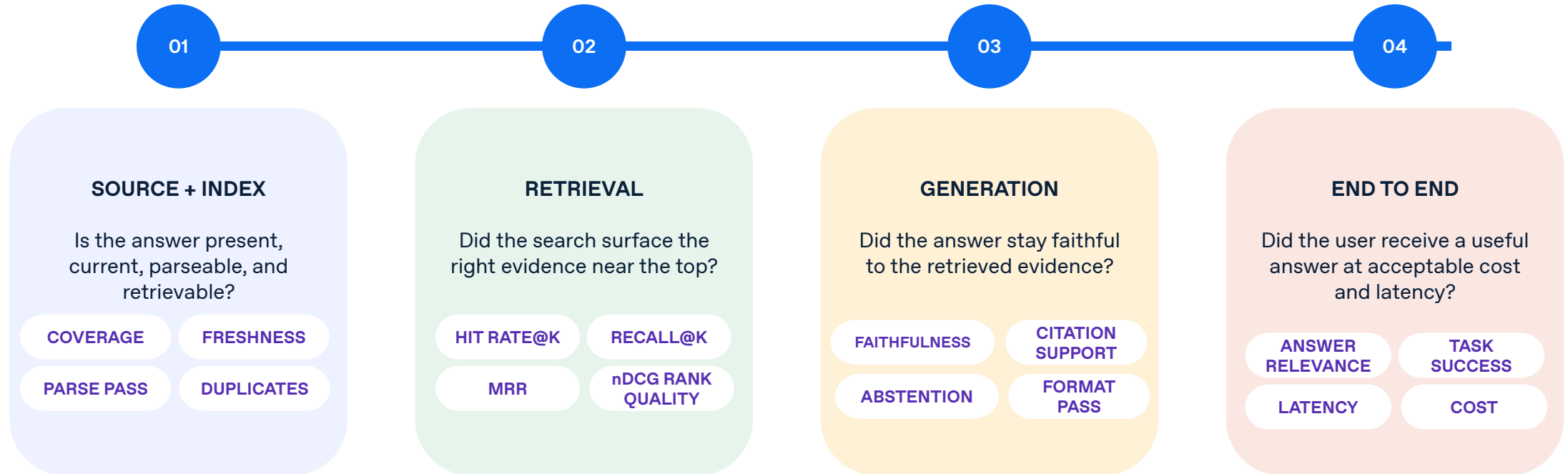
# Technical patterns

Answer generation: grounded response contract



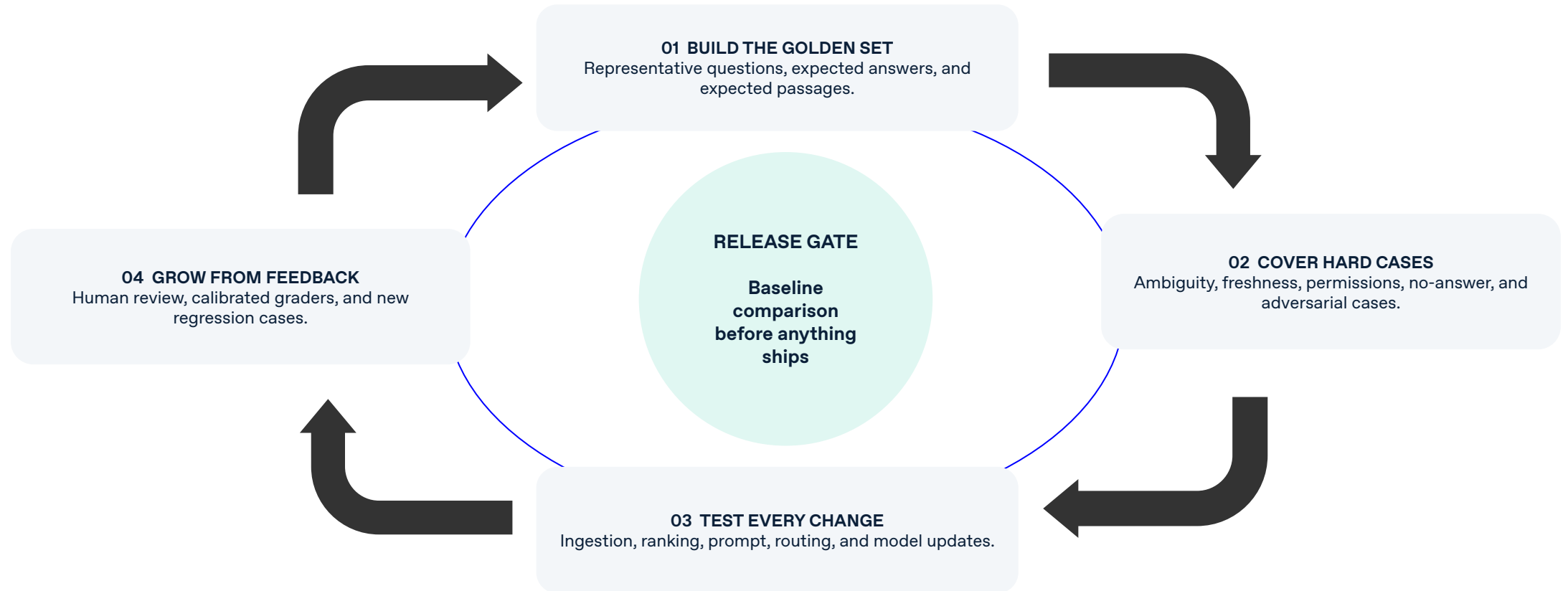
# Evaluation

Map metrics to the failing stage



# Evaluation

Build a regression loop



What can I do with this?

# Financial Services Example

## Make your data more accessible

Evaluation framework →

How can our advisors operate more effectively?

Use Cases

- Access information faster
- Reduce repetitive tasks
- Offer more specific customer insights

How can AI models support them?

Model-Level Evals

- Language translation
- Summarization of tasks
- Grading and refining content

Results →



98% of Advisors Use AI every day



Customers report higher satisfaction



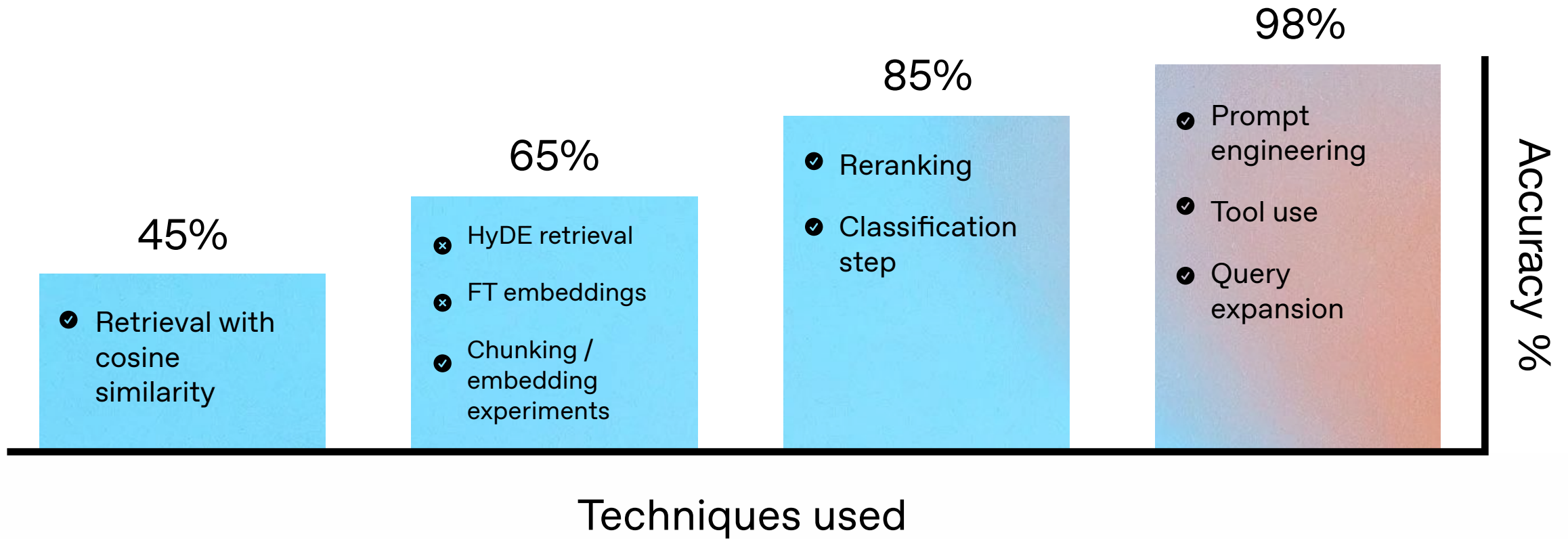
Proven evaluation framework for scaling future solutions

“

What this technology does is make you as smart as the smartest person in the organization.

- Global Head of AI

accuracy target



# Q&A

# Thank you

# Resources

- [File Search Docs](#)
- [Retrieval Docs](#)
- [Cookbook](#) - Doing RAG on PDFs using File Search in the Responses API
- [Cookbook](#) - Multi-Tool Orchestration with RAG approach using OpenAI's Responses API
- [Cookbook](#) - How to parse PDF docs for RAG
- [Cookbook](#) - How to combine GPT4o mini with RAG to create a clothing matchmaker app