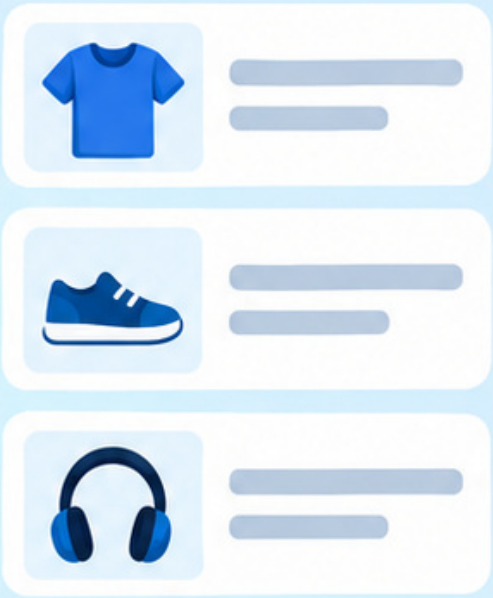
AI SYSTEM DESIGN • DAY 17

FIND CANDIDATES. THEN ORDER THEM.

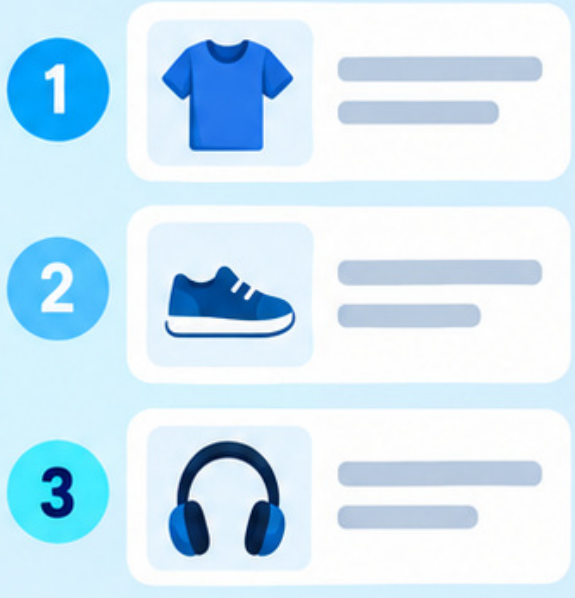
RECOMMENDATION RETRIEVAL AND RANKING

SHOP CATALOG

CANDIDATE SHORTLIST



ORDERED RECOMMENDATIONS

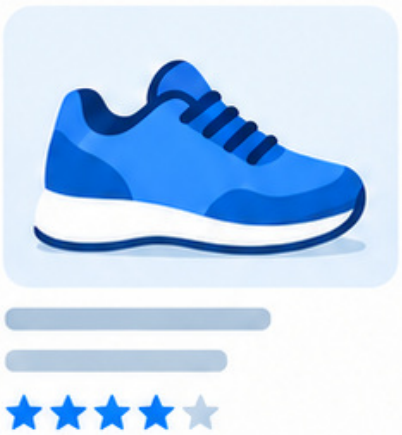


A useful recommendation must be relevant, available, and delivered in time.

Today: separate the shortlist from the final order, then measure each stage.

DEFINE THE RECOMMENDATION TASK

Illustrative task: show three related products on a product page.



Define the user context, eligible catalog, relevance label, and response budget.

Similar items and personalized items are different tasks.

Similar items (related to the current item)

Personalized items (relevant to the user)

A related-item baseline can work without a user history.

WHY USE TWO STAGES?

RETRIEVAL finds a manageable set of plausible candidates.

RANKING spends more work ordering those candidates.

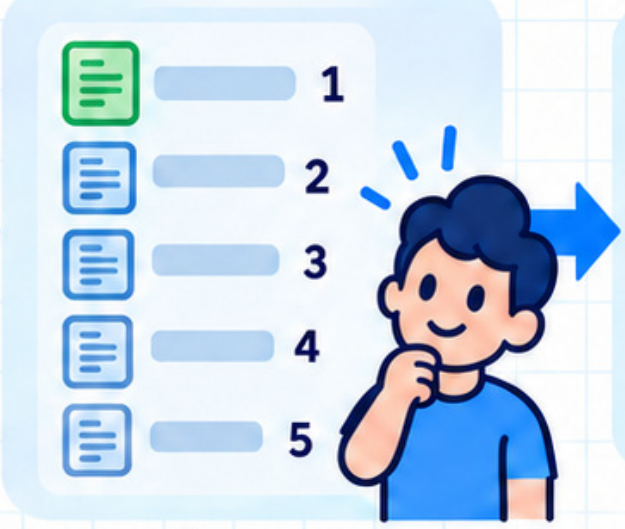
Wide catalog

Many candidates

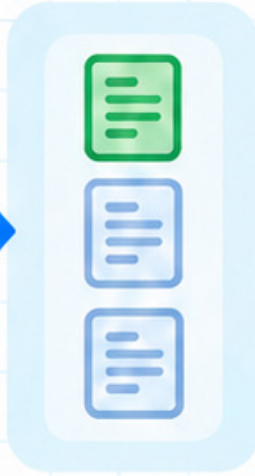
Shortlist
(from retrieval)

A manageable set of plausible candidates.

Ranking



Final list



Top results

A relevant item missed by retrieval.

A ranker cannot recover a relevant item that retrieval never returned.

The candidate count trades coverage against downstream work.

DRAW THE COMPLETE PATH

Request context

eligible candidate retrieval

ranking

final availability check

response

Availability check fails (too few items survive).

If too few items survive, refill within budget or return fewer items.

Return fewer items.

Use current permissions and catalog rules.

Recheck volatile availability before display.

A SIMPLE RETRIEVAL BASELINE

Illustrative products, vectors, and configuration

1 Represent each product with declared text or category features.

Running Shoes
Category: Sports
Features: breathable, lightweight, running

2 Use nearest neighbors to find similar products.

3 Exclude the current product and duplicate IDs.

4 Version the representation and distance choice.

Representation: v1
Distance: cosine
Features: text + category

5 Similarity is a candidate signal, not proof that the user will buy.

WHAT THE TOOLS ACTUALLY DO

scikit-learn NearestNeighbors returns nearby rows under the configured distance.

Your application maps row indices to product IDs.

Illustrative mapping

Row Index	Product ID
0	P1001
1	P1043
2	P2098
3	P3175
…	…

PostgreSQL can filter catalog records using availability and access rules.

- In stock
- Meets access rules
- Visible in catalog

Application code combines candidates, applies budgets, and handles fallback.

Combine candidates

↓

Apply budgets

↓

Handle fallback

USE MORE THAN ONE CANDIDATE SOURCE

Illustrative sources:

Similar products from item features.

Popular eligible products in the category.

Personal-history candidates when appropriate data exists.

Merge by product ID and keep source provenance.

Deduplicated candidate product IDs with sources

ID: A1	Similar, Popular
ID: B2	Popular, History
ID: C3	Similar
ID: D4	Popular, History
ID: E5	History

A source quota can improve coverage, but must be evaluated.

AI SYSTEM DESIGN • DAY 17

8/18

RANKING HAS A DIFFERENT JOB

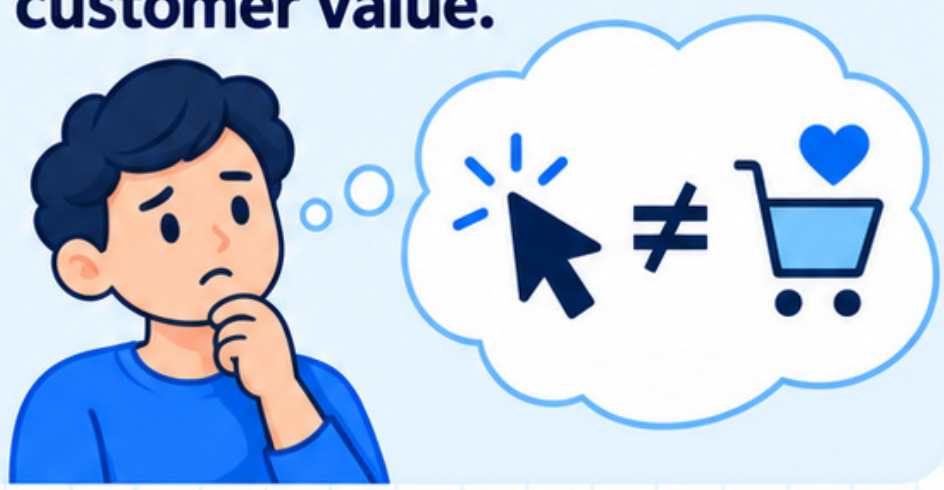
For each candidate, build features available at request time.

Estimate the declared relevance objective, then order the candidates.

A rule or simple model is a useful baseline.

- [x] Simple rules
- [x] Linear model
- [x] Tree model

A high click score does not automatically mean high customer value.

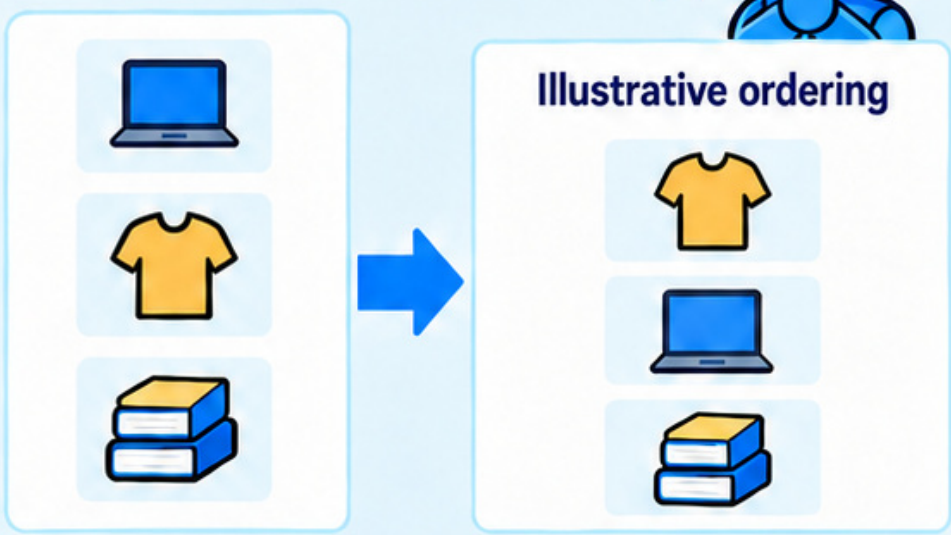


@learn.machinelearning

KEEP CONSTRAINTS EXPLICIT

Hard constraints remove ineligible items.

Soft preferences influence ordering.



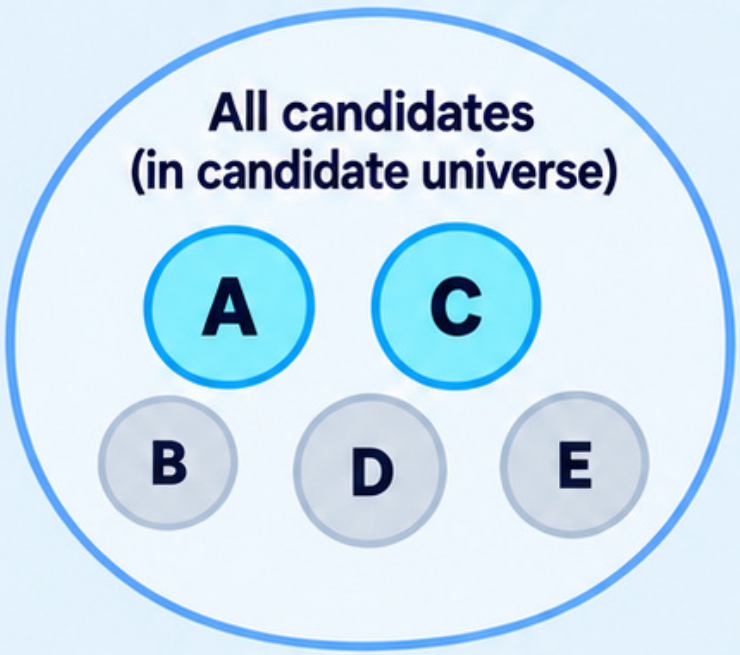
Examples: unavailable stock is a hard exclusion; category variety can be a soft preference.

Apply the declared policy in code and evaluate the final list.

Do not hide business rules inside an unexplained score.

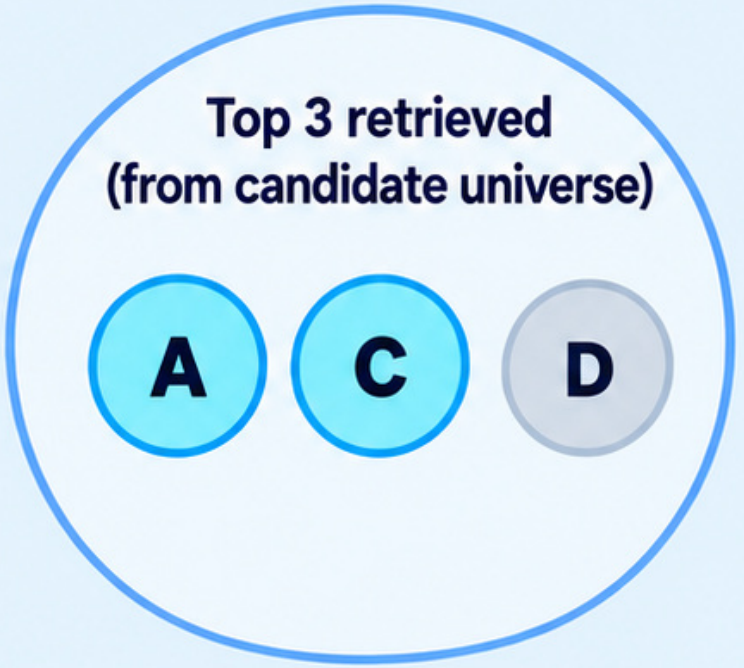
MEASURE CANDIDATE RECALL

Illustrative eligible relevant set: {A, C}.



Relevant items: {A, C}

Retrieved top 3: {A, C, D}.



Recall@3 = relevant items retrieved / all eligible relevant items.

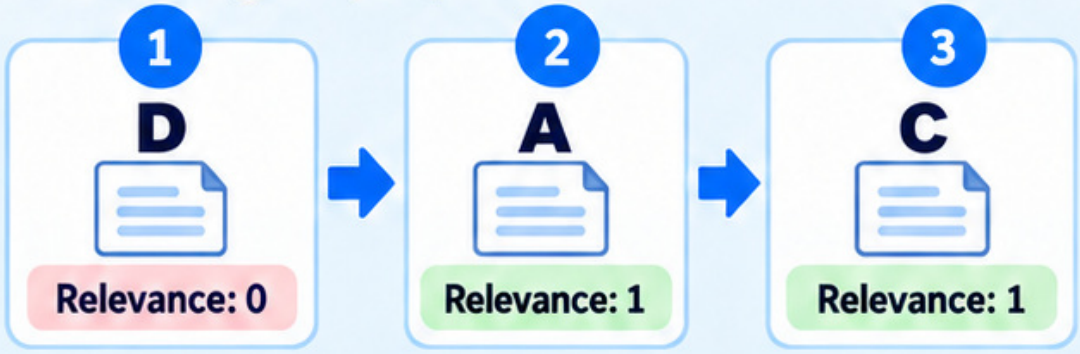
Here: 2 / 2 = 1.00.

Declare the candidate universe and relevance labels before comparing systems.

MEASURE THE FINAL ORDER

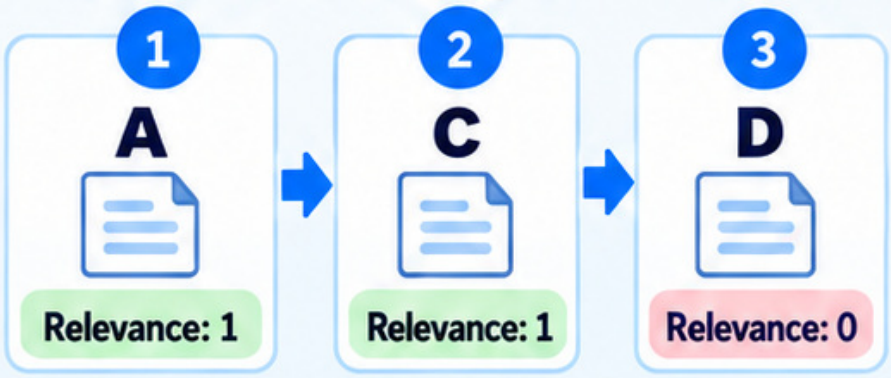
Illustrative binary relevance: A=1, C=1, D=0.

Ranking: D, A, C.



$$DCG@3 = 0 + 1/\log_2(3) + 1/\log_2(4).$$

Ideal order: A, C, D.

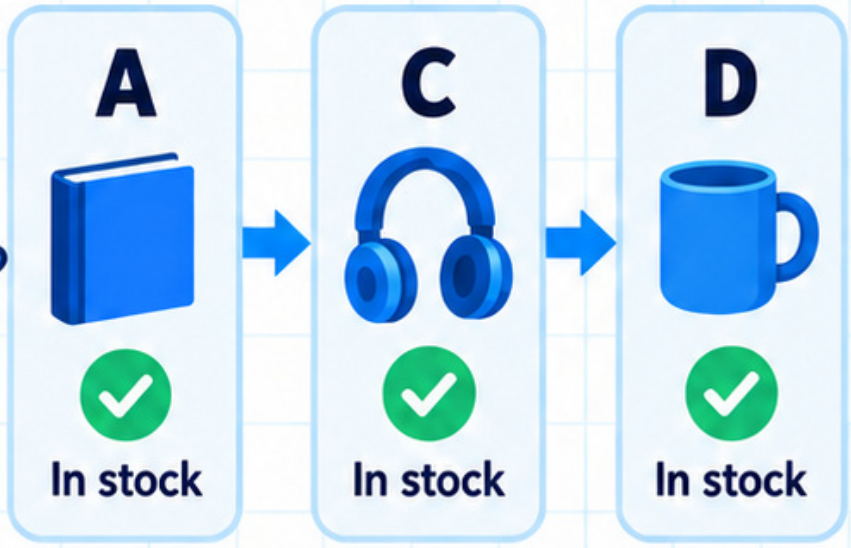


$$NDCG@3 \approx 1.131 / 1.631 \approx 0.693.$$

Relevant items earn more credit near the top.

CHECK THE SERVED LIST TOO

Illustrative final list:
A, C, D.



FINAL CHECK RESULTS

Unavailable items at the final check: 0.

Unavailable-item rate = 0 / 3 = 0%.

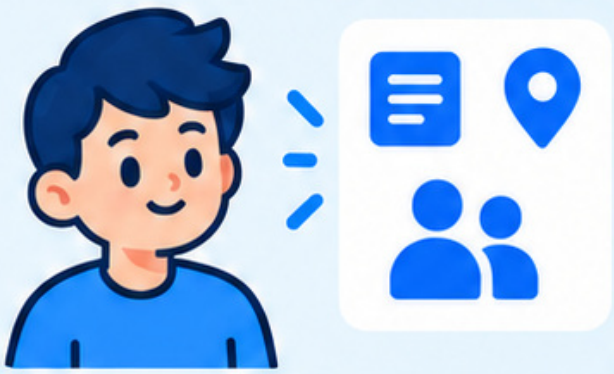
0/3
= 0%

Also measure empty responses, fallback rate, and latency.

A good ranking score cannot excuse an ineligible recommendation.

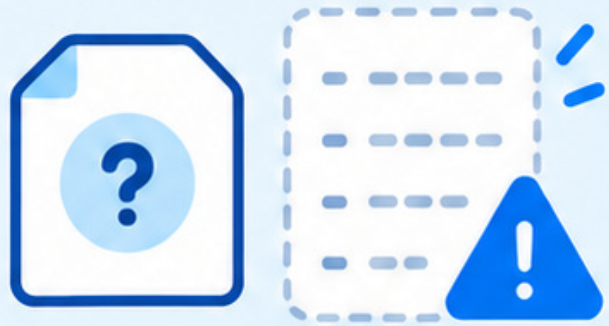
COLD START NEEDS A ROUTE

New user: use contextual or popular eligible items.



New product: use content features when available.

Missing or invalid features: use the declared fallback.



Fallback items still pass availability and permission checks.

Measure these populations separately.

AI SYSTEM DESIGN • DAY 17

14/18

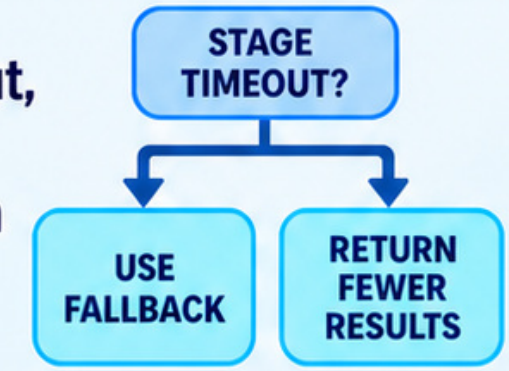
FIT THE REQUEST BUDGET

Allocate time to retrieval, feature reads, ranking, and final checks.

Measure the full request distribution, including fallbacks and failures.

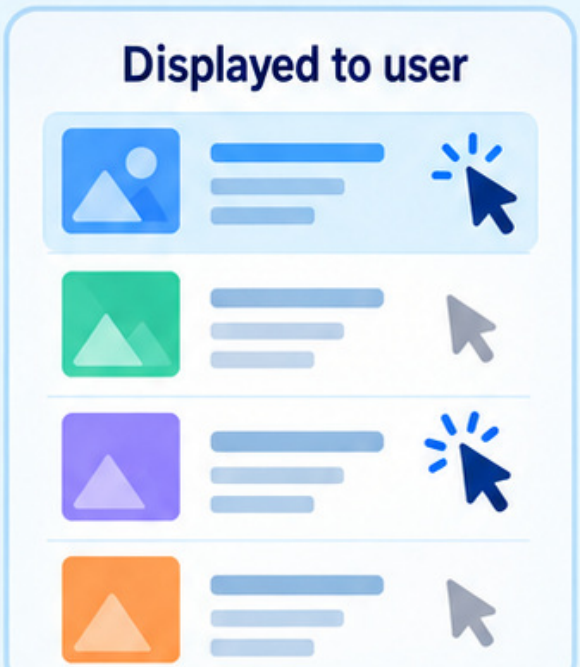
Bound candidate count and refill attempts.

If a stage times out, use the declared fallback or return fewer results.

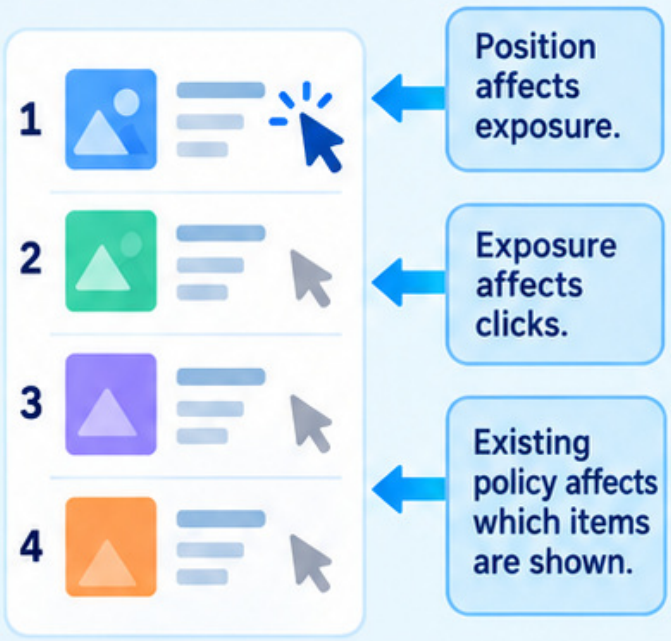


OFFLINE RELEVANCE HAS LIMITS

Clicks are observed only for displayed items.



Position, exposure, and existing policy affect those clicks.



An unclicked item is not automatically irrelevant.

Use a defined labeling protocol and realistic time splits.

Offline gains need careful online validation before business claims.

SAVE ENOUGH TO DIAGNOSE ERRORS

Log request context under an appropriate privacy policy.

Record candidate IDs and sources, feature and model versions, filter reasons, final order, and timing.

Join later feedback to the correct impression.

This separates a missing candidate from a bad ranking or availability failure.

FOLLOW ONE REQUEST

Retrieval	Ranking	Filter	Impression	Later feedback
Candidate IDs and sources	Feature and model versions	Filter reasons	Final order and timing	Join to the correct impression

YOUR TURN: LOCATE THE MISS

Relevant set

Illustrative relevant set: {A, C}.

A C

Candidates

Candidates: {A, D, E}.

A D E

Final ranking

Final ranking: D, A, E. All are available.

D A E

1. What is recall@3?
2. Which stage lost C?
3. Would a better ranker alone recover C?

EVALUATE EACH STAGE

Exercise: recall@3 = 1 / 2 = 0.50. Retrieval missed C; ranking cannot add it back.

RETRIEVAL (top 3)

- A
- D
- E

MISSED

- C ✗

Not retrieved → cannot be added back by ranking.

A ranker can move A upward, while retrieval needs its own fix.

RANKING (reorder retrieved)

Before	After
D	A
A	D
E	E

RETRIEVAL FIX (add C)

- C

Needs its own fix.

Check relevance, availability, coverage, and latency together.

- Relevance ✓
- Availability ✓
- Coverage ✓
- Latency ✓

Next: connect the complete ML serving design.