RAG: Tricks from the trenches

Practical lessons from building retrieval systems in Python

Duarte O.Carmo - duarteocarmo.com

Ciao, sono Duarte.

/du-art/ - originally from Portugal, based in Denmark

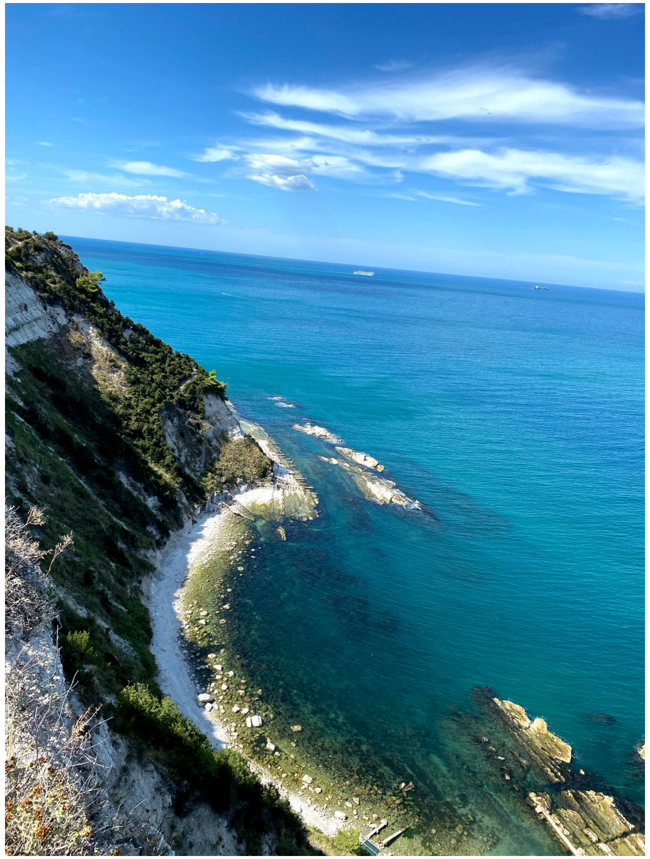
ML/Software/AI Engineer & contractor 7+ years exp.

Past: Strategy, Product Mgmt., Mgmt. Consulting

Now: I help companies solve **hard** problems

Future: Ancona/Le Marche (convincing wife still)

I have a blog/newsletter – you might like it



Today, we'll talk about RAG.

Simple and practical advice.

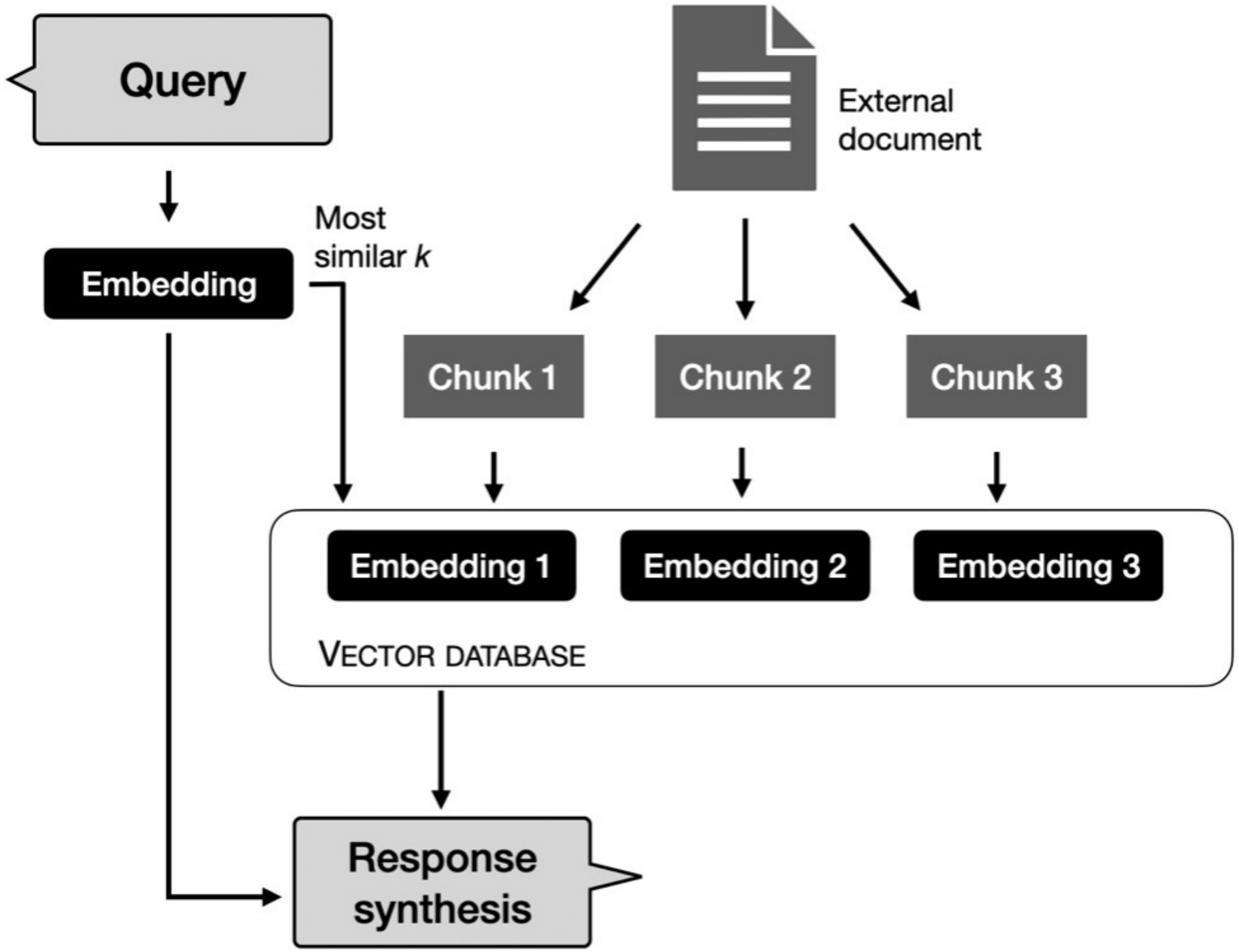
1. What's RAG?
2. Your embedding choice sucks
3. Structured outputs matter
4. Exploding contexts and summaries
5. Your beautifully ugly production data
6. Is your system actually any good?
7. Final thoughts and resources for beyond

1.What’s RAG?

```
duarteocarmo@Duartes-MacBook-Pro:~|master⚡*
➤ ollama run qwen3:30b-a3b
>>> where is pycon italia this year? /no_think
<think>

</think>

As of now (2024), **PyCon Italia** is typically held in **Rome**, but the exact venue and
date can vary each year. For the most accurate and up-to-date information about **PyCon
Italia 2024**, I recommend checking the official website or their social media channels.

You can visit the official PyCon Italia website at:
[https://pycon.it/](https://pycon.it/)

They usually announce the location, date, and program there. If you’re looking for a
specific year, let me know, and I can help you find the details!
```

RAG = provide relevant context to LLM*

*) Hamel Husain



Source: Sebastian Raschka – Machine Learning Q and AI

RAG != Vector DB

```python
from sentence_transformers import SentenceTransformer, util

model = SentenceTransformer("all-MiniLM-L6-v2", device="mps")

corpus = [
    "Talk on large language models and production.",
    "How to deploy Python apps with Docker.",
    "Tips for writing better Python tests.",
    "Conference opening with keynote speaker.",
]

query = "Deploying Python with containers"
corpus_embeddings = model.encode(corpus, convert_to_tensor=True)
query_embedding = model.encode(query, convert_to_tensor=True)

hits = util.semantic_search(query_embedding, corpus_embeddings, top_k=1)[0][0]
score = hits["score"]
sentence = corpus[hits["corpus_id"]]
print(f"Most similar: {sentence}")
print(f"Score: {score:.4f}")
# Most similar: How to deploy Python apps with Docker.
# Score: 0.850
```

```python
import lancedb
from lancedb.pydantic import LanceModel, Vector
from lancedb.embeddings import get_registry

MODEL = "BAAI/bge-small-en-v1.5"
DEVICE = "mps"
DB_PATH = "/tmp/db"
TABLE_NAME = "words"

db = lancedb.connect(DB_PATH)
model = get_registry().get("sentence-transformers").create(name=MODEL, device=DEVICE)

class Words(LanceModel):
    text: str = model.SourceField()
    vector: Vector(model.ndims()) = model.VectorField()

if TABLE_NAME not in db.table_names():
    table = db.create_table("words", schema=Words)
    data = [{"text": "I live in Ancona"}, {"text": "I was born in Rome"}]
    table.add(data)

table = db.open_table(TABLE_NAME)

query = "Where was I born?"
result = table.search(query).select(["text"]).limit(1).to_list()
text_results = [r["text"] for r in result]

prompt = f"""
You are a helpful assistant. Answer the question based on the context provided.
Context:
{text_results}
Question: {query}
""".strip()
call_llm(prompt)
```

RAG has become *simple*...

But has it *really*?

Why did you choose those embeddings?

Are users ever going to trust your responses?

What if users ask for a summary of you 10GB table?

Is your prod data really that clean? (my Google's aren't)

How do we know our system is any good?

2.Your embedding choice sucks

Consider a simple example.

```
user_query = ”Can weight affect asthma?”
```

Search this dataset

_id string · *lengths* 36 – 36	text string · *lengths* 39 – 4.77k	title string · *lengths* 8 – 706
00022521-710c-1106-5099-2c7bffe70e7d	However, respondents were generally not familiar with the concept of contact tracing or the…	Primary health care staff's perceptions of childhood tuberculosis: a qualitative study from…
0002edd2-74bf-1db5-578b-193d63edb651	We also explored the impacts FIC trainees had on health policy in their countries and globally…	The impact of Fogarty International Center research training programs on public health…
00035a34-85e4-6d6f-3360-418d795cf2a9	Our paper is the first to report correlations of the abdominal wall measurements with fetal age,…	Investigation of a connection between abdominal wall defects and severity of the herniation in…
0003d86f-96c9-20f0-99fa-197efb6f4478	Results from trial 2 with CR rabbits were not always duplicated in this comparative trial.…	Divergent Evolutionary Pathways of Myxoma Virus in Australia: Virulence Phenotypes in Susceptibl…
000690b3-0782-5b83-b33f-4fdac4b3c5d2	The use of medical information technology has become an important way to improve the level of…	Obstructive Sleep Apnea Syndrome Treated Using a Positive Pressure Ventilator Based on Artificial…
000762b2-da65-c4ab-3043-5f9715baf7ff	It is evident from the RTE verdicts that it does not require patients to meet all four Appelbaum…	Euthanasia and assisted suicide for people with an intellectual disability and/or autism spectru…
00078214-8a80-3d6b-404c-0e27d3fe03ca	Conclusions: PPT showed good correlation with systolic blood pressure and may have potential…	38th International Symposium on Intensive Care and Emergency Medicine Brussels, Belgium. 20-23…
00138824-abce-c0e9-6038-ad6a7eb291ea	The aim of this study is to comprehensively evaluate the impact of Dupilumab on olfactory…	Olfaction Recovery following Dupilumab Is Independent of Nasal Polyp Reduction in CRSwNP
001832fd-bdaf-d3f1-4d3d-894bbf3fe132	We first examined descriptive statistics for each variable and formally tested sex…	Do the associations of body mass index and waist circumference with back pain change as people…
0018bb80-95de-857a-82ac-d015fa89c695	Table 1 Baseline characteristics of SHARP patients included in the cost analysis	What is the impact of chronic kidney disease stage and cardiovascular disease on the annual…

32K passages from open-access medical articles

Source: Hugging Face

```python
from sentence_transformers import SentenceTransformer, util
import torch

# Define the user query
user_query = "Can weight affect asthma?"

# Sample corpus of documents
corpus = [
 # our 32K medical passages
]

# Load a pre-trained Sentence Transformer model
model = SentenceTransformer('all-MiniLM-L6-v2')

# Encode the corpus and the query
corpus_embeddings = model.encode(corpus, convert_to_tensor=True)
query_embedding = model.encode(user_query, convert_to_tensor=True)

# Perform semantic search to find the top 10 most similar documents
top_k = 10
hits = util.semantic_search(query_embedding, corpus_embeddings,
top_k=top_k)[0]

# Display the results
print(f"Query: {user_query}\n")
print("Top 10 most similar documents:")
for hit in hits
    print(f"{corpus[hit['corpus_id']]} (Score: {hit['score']:.4f})")
```

Retrieve the most similar 10

For each result check relevancy

We score around 16.54%*

Now we change **1 line**

* - Normalized discounted cumulative gain

```python
from sentence_transformers import SentenceTransformer, util
import torch

# Define the user query
user_query = "Can weight affect asthma?"

# Sample corpus of documents
corpus = [
 # our 32K medical passages
]

# Load a pre-trained Sentence Transformer model
model = SentenceTransformer('nvidia/NV-Embed-v1')

# Encode the corpus and the query
corpus_embeddings = model.encode(corpus, convert_to_tensor=True)
query_embedding = model.encode(user_query, convert_to_tensor=True)

# Perform semantic search to find the top 10 most similar documents
top_k = 10
hits = util.semantic_search(query_embedding, corpus_embeddings,
top_k=top_k)[0]

# Display the results
print(f"Query: {user_query}\n")
print("Top 10 most similar documents:")
for hit in hits
    print(f"{corpus[hit['corpus_id']]} (Score: {hit['score']:.4f})")
```

Retrieve the most similar 10

For each result check relevancy

We score around ~~16.54%~~ **62.89%**

That's a **~281.76%** increase.

Cure V1 dataset (Open on HF)

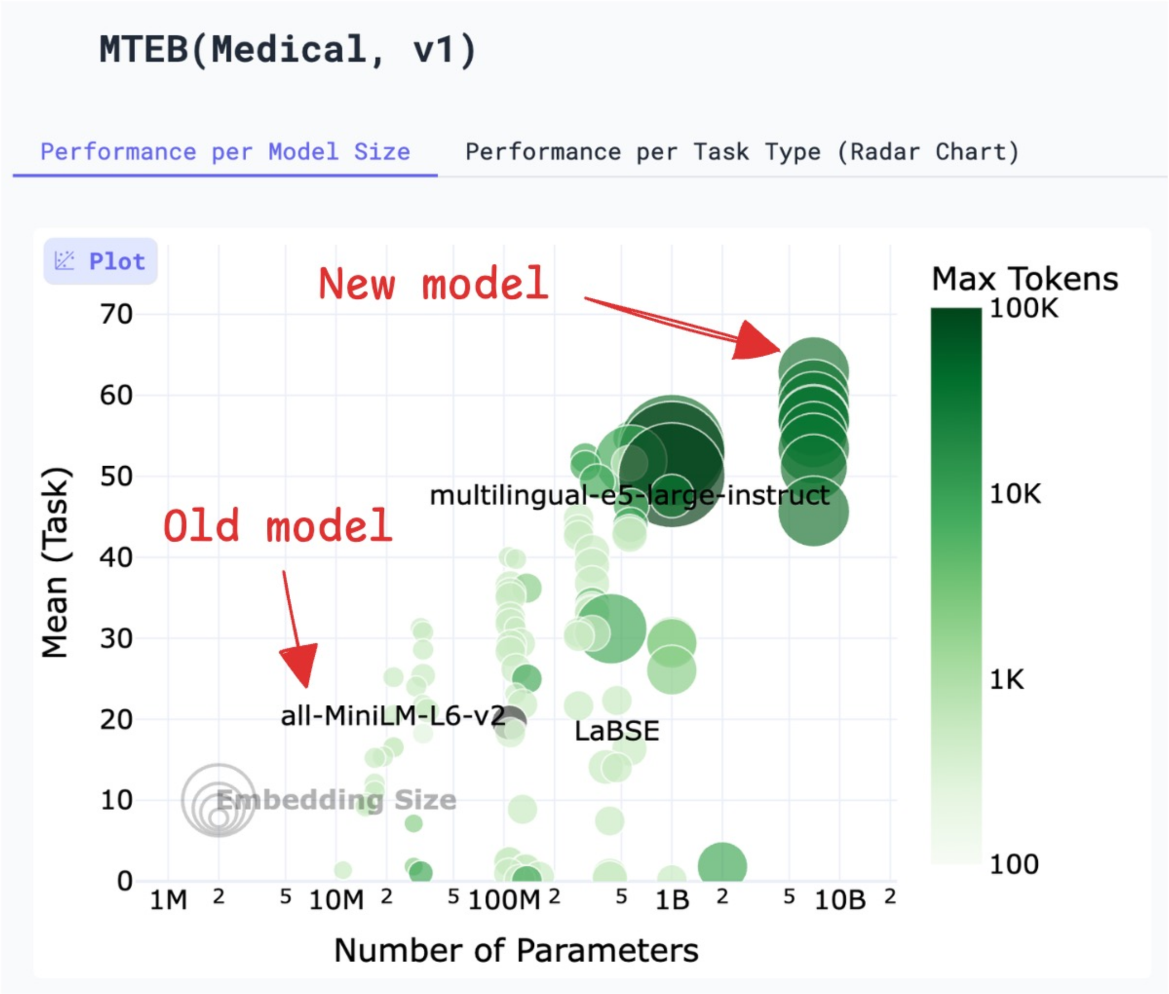
MTEB leaderboard

Multiple languages

Different domains (medical, code)

Tasks (Retrieval, Clustering..)

All open source!



Source: MTEB: Massive Text Embedding Benchmark

Embeddings matter..

But <u>nothing</u> comes for free..!

3.Structured outputs matter

```python
import instructor
from openai import OpenAI
from pydantic import BaseModel

# Define your output structure
class UserInfo(BaseModel):
    name: str
    age: int

# Create an instructor-patched client
client = instructor.from_openai(OpenAI())

# Extract structured data
user_info = client.chat.completions.create(
    model="gpt-3.5-turbo",
    response_model=UserInfo,
    messages=[
        {"role": "user", "content": "John Doe is 30 years old."}
    ],
)

print(f"Name: {user_info.name}, Age: {user_info.age}")
# Output: Name: John Doe, Age: 30
```

Bringing sanity to LLM apps

Integrating with rest of world

This *is* MCP

<u>Many</u> libraries

Source: Instructor documentation (adapted)

Users have trust issues with LLMs.

Learning: Make verification easy.

```python
class Fact(BaseModel):
    fact: str
    substring_quote: List[str]

    @model_validator(mode="after")
    def validate_sources(self, info: ValidationInfo) -> "Fact":
        ctx = info.context["text_chunk"]
        self.substring_quote = [
            ctx[s:e] for q in self.substring_quote
            for s, e in re.finditer(re.escape(q), ctx)
        ]
        return self

class QuestionAnswer(BaseModel):
    question: str
    answer: List[Fact]

    @model_validator(mode="after")
    def validate_sources(self) -> "QuestionAnswer":
        self.answer = [f for f in self.answer if f.substring_quote]
        return self
```

Every Fact is supported

Every Fact is validated

Pydantic validators for the win

Source: Instructor documentation (adapted)

```python
question = "What does Duarte do?"
context = """
Hi! My name is Duarte. I’m a Portuguese technologist living in Copenhagen.
I’m attending PyCon Italia in Bologna to speak about LLMs, validation, and reliable citations.
I also founded a startup that uses AI to analyze technical documents.
"""

ask_ai(question, context)

# Response
{
  "question": "What does Duarte do?",
  "answer": [
    {
      "fact": "Duarte is a Portuguese technologist living in Copenhagen.",
      "substring_quote": ["Portuguese technologist living in Copenhagen"]
    },
    {
      "fact": "He’s attending PyCon Italia in Bologna to speak about LLMs.",
      "substring_quote": ["PyCon Italia in Bologna"]
    }
  ]
}
```

Source: Instructor documentation (adapted)

Are all questions the same?

API calls..

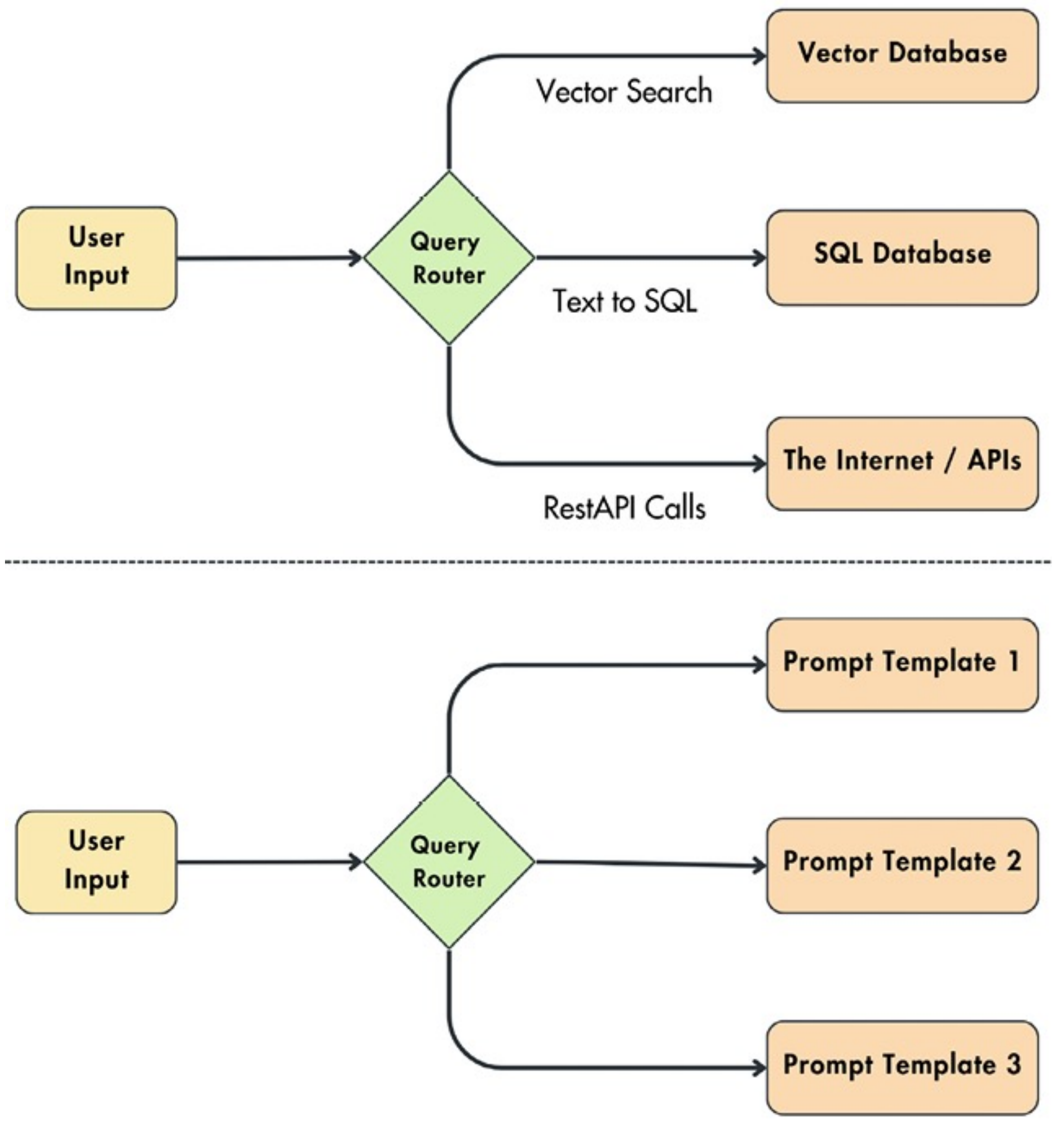
Different prompts..

Different escalations..

Human-in-the-loop..

Questions are not all created equal.

```python
class QueryType(str, Enum):
    VECTOR_SEARCH = "vector_search"
    TEXT_TO_SQL = "text_to_sql"
    REST_API = "rest_api"


class StructuredQuery(BaseModel):
    question: str
    query_type: QueryType
```



Source: LLM Engineer's playbook

4.Exploding contexts and summaries

Imagine a database of 2 Million skeets*

```
df = (
    pl.scan_ndjson("hf://datasets/alpindale/two-million-bluesky-posts/*.jsonl")
    .filter(pl.col("reply_to").is_null())
    .collect()
)
```

*) It's 🦋 for tweets

”What are the main trends?”

“What are people saying about Bologna?”

Should we treat these equally?

Representative filter

Keyword filter

More general ← LLM context → More specific

"What are the trends regarding GPU computing?"

"Summarize the dataset"

"What are the main trends?"

We want to give the LLM a 'general' feel

How do we do this?

"How's the weather in Lisbon?"

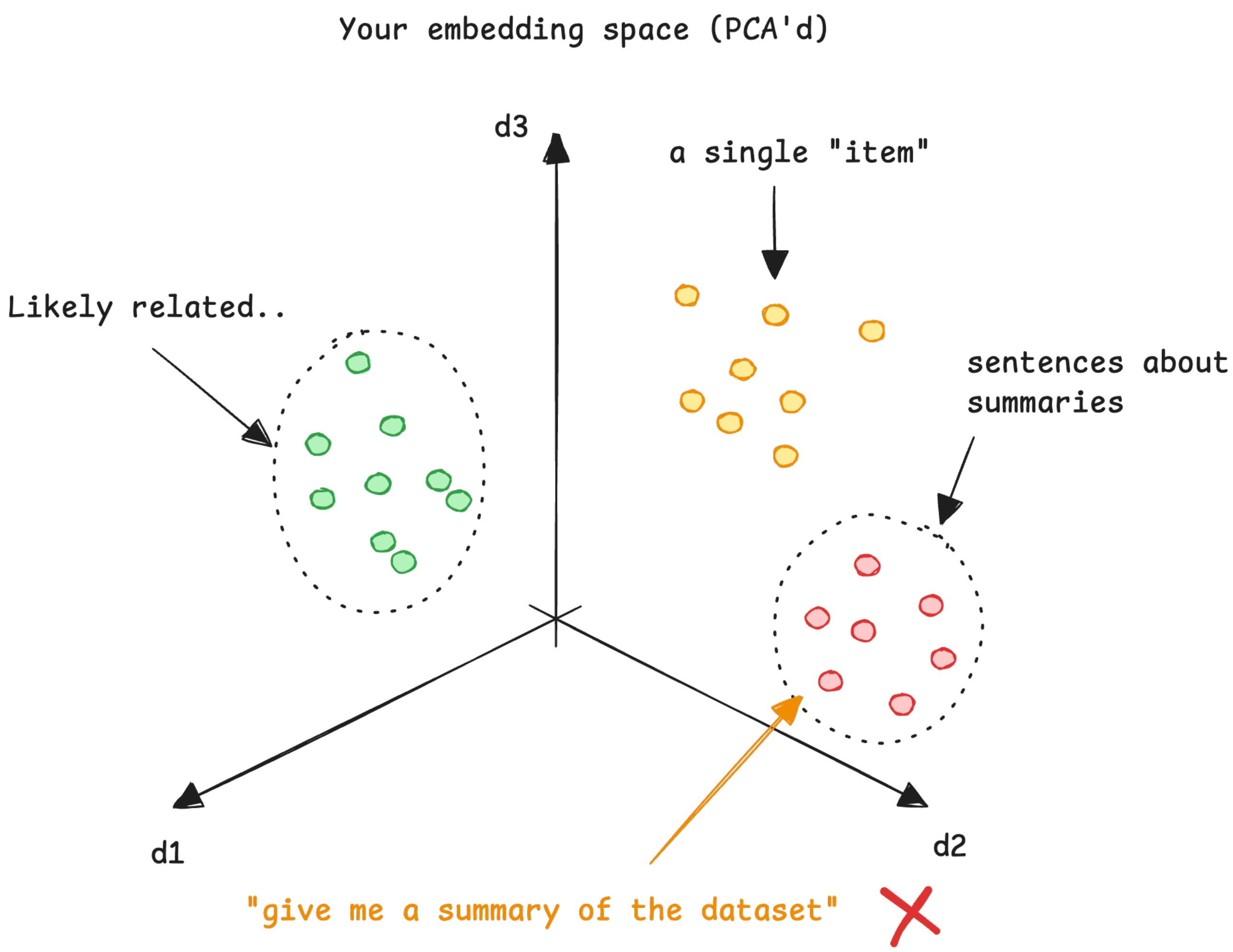
"Why are people leaving BlueSky?"

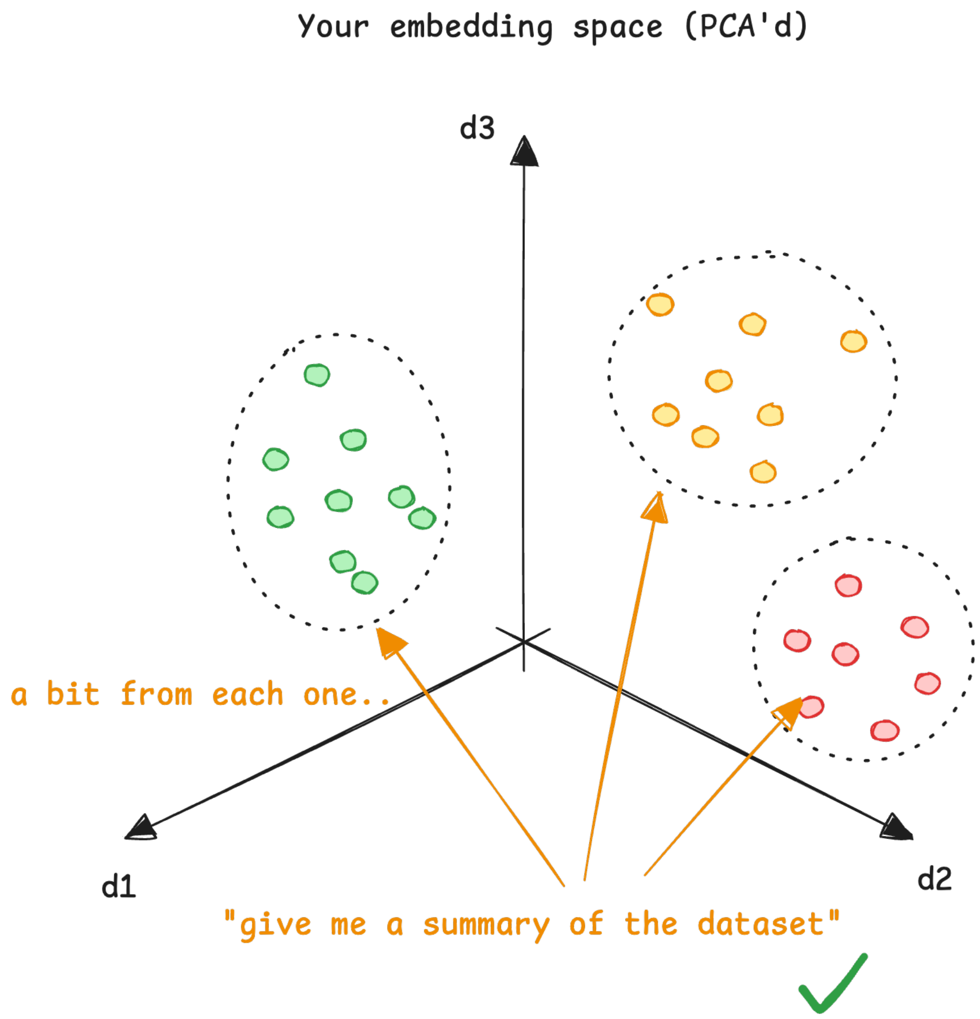
We want to give the LLM specific context

RAG (get nearest neighbours)

```python
def get_intent(question: str):
    INTENT_PROMPT = f"""
    Tell me which filter we should use for the given user question.
    The keyword filter filters data based on keywords from the question. This is good for specific
questions or when you want to focus on a particular topic.
    The representative filter returns a representative sample of insights, from which you can infer the
answer to the question. This is good for questions that relate to the entire dataset (trends, summaries,
etc)
    Respond only with the type of filter to use!

    Examples:
    'Summarize the data' -> Representative filter
    'What are the main insights?' -> Representative filter
    'What are the high-level discussion points from our field reps regarding NOS ?' -> Keyword filter
    'Describe the negative, neutral and positive perception of physicians for Benuron?' -> Keyword filter
    'What are the main trends from the following skeets?' -> Representative filter
    'What are the main trends?' -> Representative filter
    'What are the key tweets I should be aware of?' -> Representative filter

    Question: {question}
    """.strip()
    return llm(INTENT_PROMPT)
```

Example query routing


Your embedding space (PCA'd)
d3
a single "item"
Likely related..
sentences about summaries
d1
d2
"give me a summary of the dataset"


Your embedding space (PCA'd)
d3
a bit from each one..
d1
d2
"give me a summary of the dataset"

```python
# create clusters
kmeans = KMeans(n_clusters=n_clusters, random_state=42)
embeddings = [_.vector for _ in results_list]
kmeans.fit(embeddings)

cluster_labels = kmeans.labels_

context_items = []

# get a items_per_cluster from each cluster
for cluster_id in list(set(cluster_labels)):
    cluster_items = [
        item
        for index, item in enumerate(results_list)
        if cluster_labels[index] == cluster_id
    ]
    assert len(cluster_items) == list(cluster_labels).count(cluster_id)
    context_items.extend(random.choices(cluster_items, k=items_per_cluster)

return context_items
```

KMeans

class **sklearn.cluster.KMeans**(*n_clusters=8*, ***, *init='k-means++'*, *n_init='auto'*, *max_iter=300*, *tol=0.0001*, *verbose=0*, *random_state=None*, *copy_x=True*, *algorithm='lloyd'*) [source]

K-Means clustering.

Read more in the User Guide.

Parameters:

n_clusters : *int, default=8*

The number of clusters to form as well as the number of centroids to generate.

For an example of how to choose an optimal value for `n_clusters` refer to Selecting the number of clusters with silhouette analysis on KMeans clustering.

init : *{'k-means++', 'random'}, callable or array-like of shape (n_clusters, n_features), default='k-means++'*

Method for initialization:

- 'k-means++' : selects initial cluster centroids using sampling based on an empirical probability distribution of the points' contribution to the overall inertia. This technique speeds up convergence. The algorithm implemented is "greedy k-means++". It differs from the vanilla k-means++ by making several trials at each sampling step and choosing the best centroid among them.
- 'random': choose `n_clusters` observations (rows) at random from data for the initial centroids.

5.Your beautifully ugly production data

What we think will happen

”Is treatment of class II malocclusion recommended during childhood?”

“What is the predictability of upper molar distalization with clear aligners?”

What we think will happen vs. what happens in production

”Is treatment of class II malocclusion recommended during childhood?”

Treatment malocclusion childhood ok

“What is the predictability of upper molar distalization with clear aligners?”

Distalization predictable upper molar

Query understanding

文A 1 language

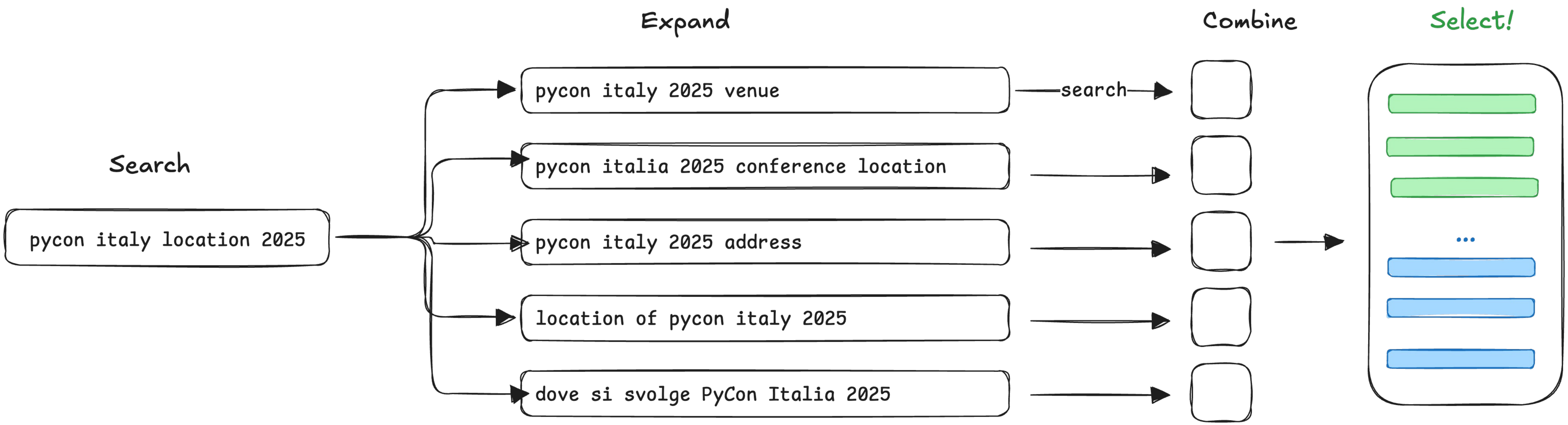
Article Talk Read Edit View history Tools

From Wikipedia, the free encyclopedia

Query understanding is the process of inferring the intent of a search engine user by extracting semantic meaning from the searcher's keywords.[1] Query understanding methods generally take place before the search engine retrieves and ranks results. It is related to natural language processing but specifically focused on the understanding of search queries.

Source: You know this one come on.

One example: Expanding queries



Derive metadata from query

Create a plan of queries from a single query

Preprocess query before search

Re-rank results (without complicating)

6.Is your system actually any good?

"The best way to get clients is to have clients."

Weinberg, Gerald - GeraldM.Weinberg-TheSecretsofConsulting_AGuidetoGivingandGettingAdviceSuccessfully

Added on Monday, 4 November 2024 17:07:27

Enjoying these? **Consider supporting the project**

Source: https://kindle-highlights.email/

~~"The best way to get clients is to have clients."~~

Weinberg, Gerald - GeraldM.Weinberg-TheSecretsofConsulting_AGuidetoGivingandGettingAdviceSuccessfully

Added on Monday, 4 November 2024 17:07:27

Enjoying these? [Consider supporting the project]

The best way to evaluate your system is to get users.

```
from ragas import evaluate

>>> dataset
Dataset({
    features: ['question', 'ground_truth', 'answer', 'contexts'],
    num_rows: 30
})

>>> result = evaluate(dataset)
>>> print(result)
{'context_precision': 0.817,
'faithfulness': 0.892,
'answer_relevancy': 0.874}
```

We all love see number go up when evaluating

Even when we have no idea what it’s doing

1. Start simple stupid

```python
# uv pip install rank-bm25
from rank_bm25 import BM25Okapi

corpus = [
    "Hello there good man!",
    "It is quite windy in London",
    "How is the weather today?"
]

tokenized_corpus = [doc.split(" ") for doc in corpus]

bm25 = BM25Okapi(tokenized_corpus)
```

2. Log everything

```python
# uv pip install rank-bm25 loguru
from loguru import logger
from rank_bm25 import BM25Okapi

corpus: list[str] = [
    "Hello there good man!",
    "It is quite windy in London",
    "How is the weather today?",
]

tokenized_corpus: list[list[str]] = [doc.split(" ") for doc in corpus]

bm25_params: dict[str, float] = {"k1": 1.5, "b": 0.75, "epsilon": 0.25}
bm25 = BM25Okapi(tokenized_corpus, **bm25_params)

query: str = "windy London"
tokenized_query: list[str] = query.split(" ")

context: list[str] = bm25.get_top_n(tokenized_query, corpus, n=5)

logger.info("Query: {}", query)
logger.info("Context: {}", context)
logger.info("BM25 params: {}", bm25_params)

response = llm(quey, context)
logger.info(response)
```

3. Improve it

You can see what is going on

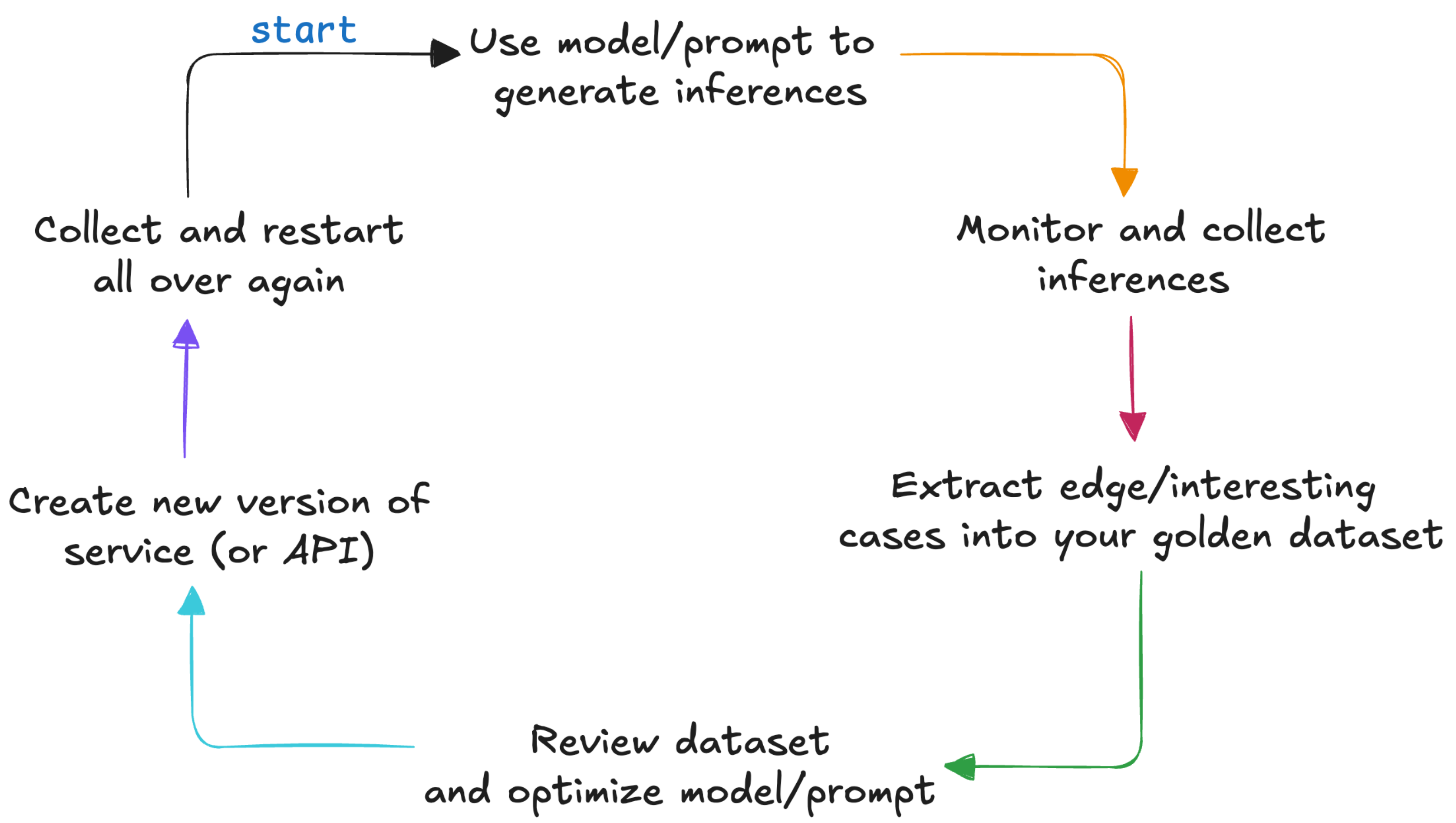
Look at your data – does it make sense?

Analyze different examples – are they any good?

Is this context relevant?*

Note: You probably care about precision.

The prompt/model evaluation and improvement fly-wheel



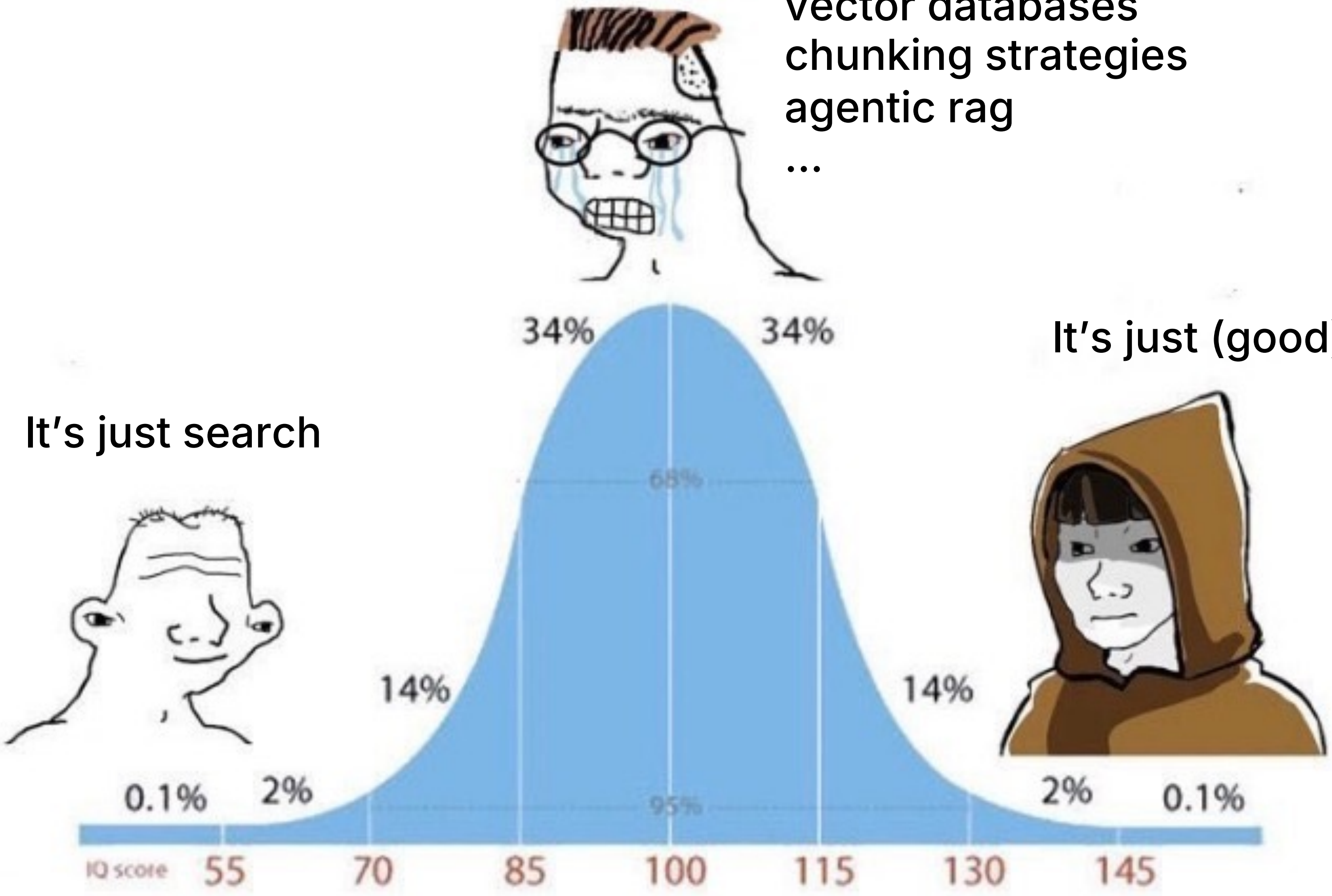
Source: duarteocarmo.com

7.Final thoughts and resources

re-ranking
fine-tuned embeddings
faithfulness, relevancy
multi-hop querying
vector databases
chunking strategies
agentic rag
...

It’s just (good) search

It’s just search



Closing the loop early and starting simple

Log the most important metrics and decisions

Be close to production

Don’t be scared to understand what is going on

Note down the issues and improve them

Repeat.

Resources

Most things by Jason Liu are great

Arcturus labs has some good writing on the topic

Simon Willison's entries are always worth a skim/read

LLM Engineer's handbook from Paul Iusztin and Maxime Labonne

Relevant search O’reilly

Handling vision in RAG

This talk by Ben Clavié

Most of these talks from Parlance labs

Evaluate rag with RAGAS

This blog post by Goku Mohandas

This talk by Sam Partee

tinyurl.com/ragresources

Grazie!