Back to basics?

Jo Kristian Bergum (@jobergum)

About

- Distinguished Engineer at vespa.ai
- *Vespa.ai is a mature serving platform spun out of Yahoo*

This talk

- Stuffing text into the LLM prompt
- Information retrieval - the R in RAG
- Evaluation of IR systems
- Building your own evals to impress your CTO
- Representational approaches for IR
- The baselines

Retrieval Augmented Generation (RAG)

Current date is {**date**}, Don’t be rude. I’ll tip $5. Think step-by-step.

I want you to classify the text input as positive, negative or neutral. Examples:

Input: I’m very happy today
Output: positive

Input: I’m sad today
Output: negative

Input: I don’t know what to feel today
Output: neutral

✨**{many_retrieved_context_sensitive_examples}**✨

Input: **{input}**
Output 🪄:

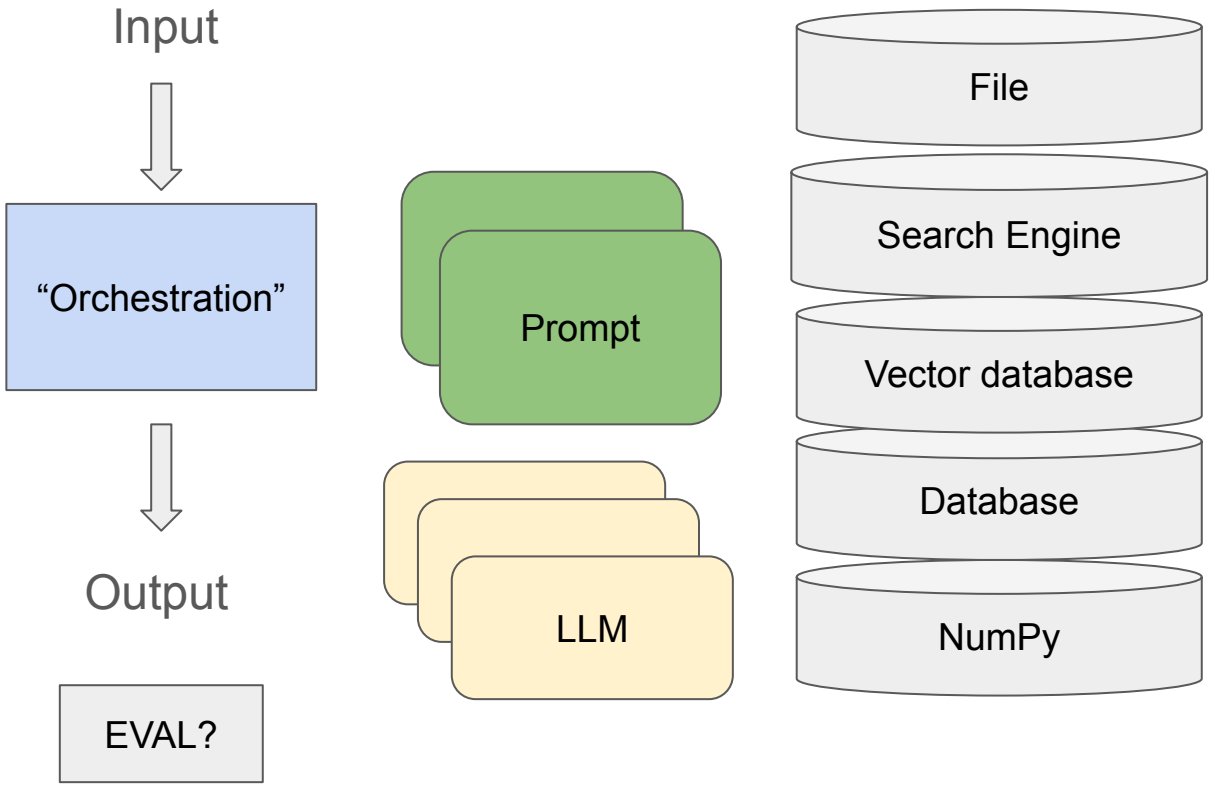
Retrieval Augmented Generation (RAG)

Current date is {**date**}, Don’t be rude. I’ll tip $5. Think step-by-step.

I want you to summarize and answer the question using context retrieved by a search engine.
Context: [1] BERT: Pre-training of deep bidirectional transformers for language understanding…
Question: What is a bidirectional transformer model?
Helpful answer: bidirectional means that tokens attend to all other tokens in the input sequence [1].
✨**{retrieved_context_sensitive_examples}**✨

Context: **{retrieved_context_question}**
Question: **{question}**
Helpful answer🪄:

Working with LLMs: Reference architecture



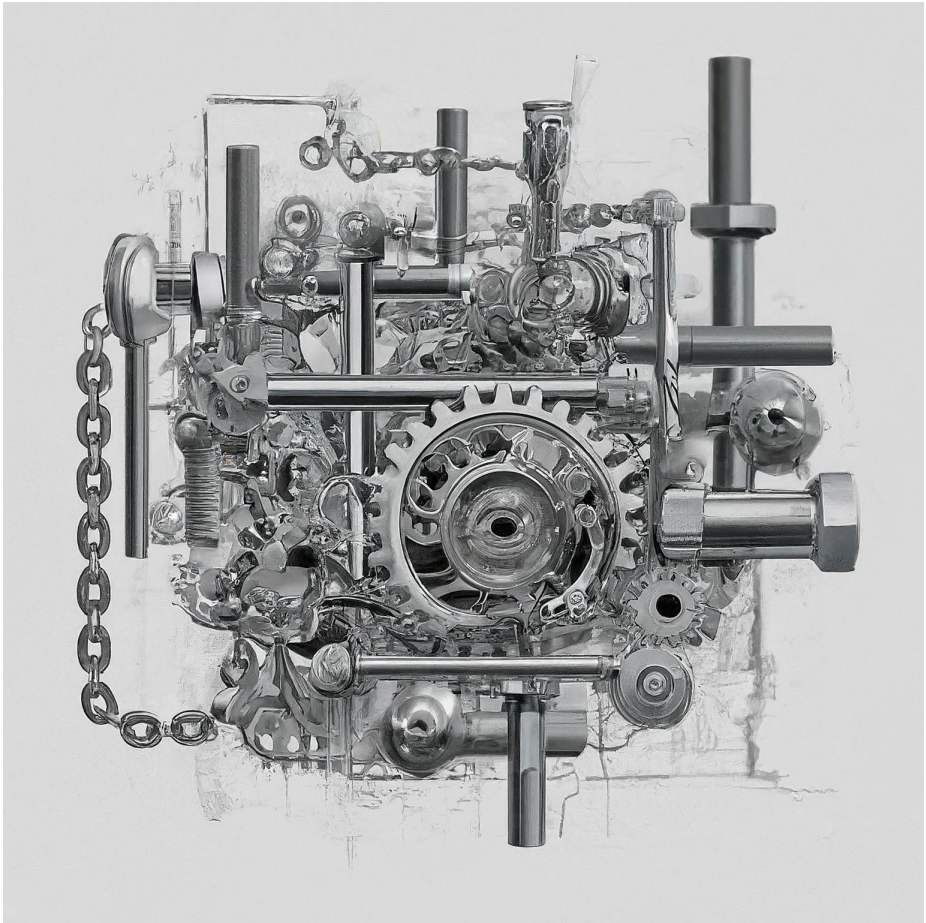
New model just dropped 🧵

Improve RAG with this weird trick 🧵

New advanced RAG retriever just dropped, here is what you need to know 🧵

New 7B embedding model with 8K context just dropped 🧵

3072 floats is all you need! 🧵

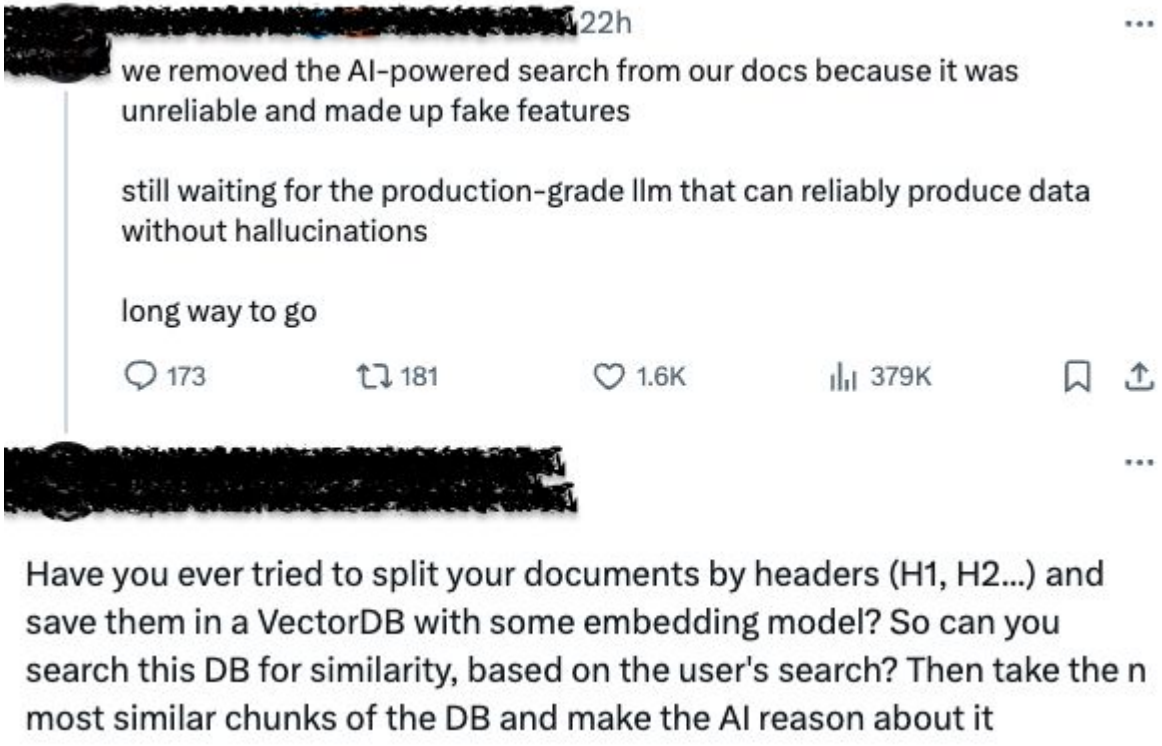


State of RAG 2024

RAG

LLM + “AI” Vectors from

Call it a day?

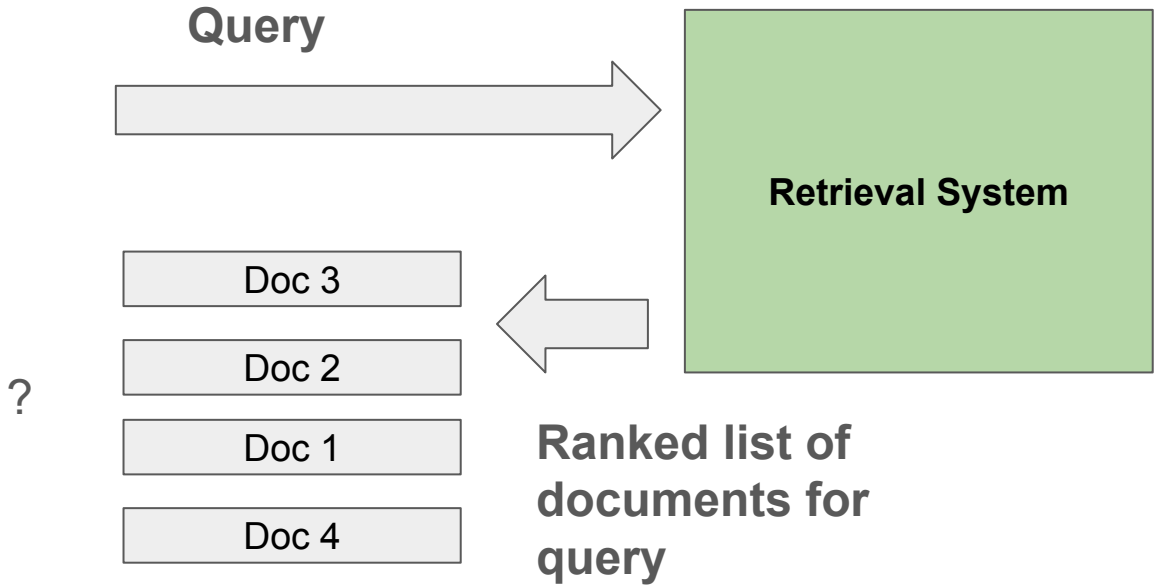


The R in RAG: Information Retrieval (IR)

- *The process of obtaining relevant information based on a user's information need expressed as a query/question*

Occupied the brightest minds in computer science for decades

Evaluating Information Retrieval Systems



Evaluating Information Retrieval Systems

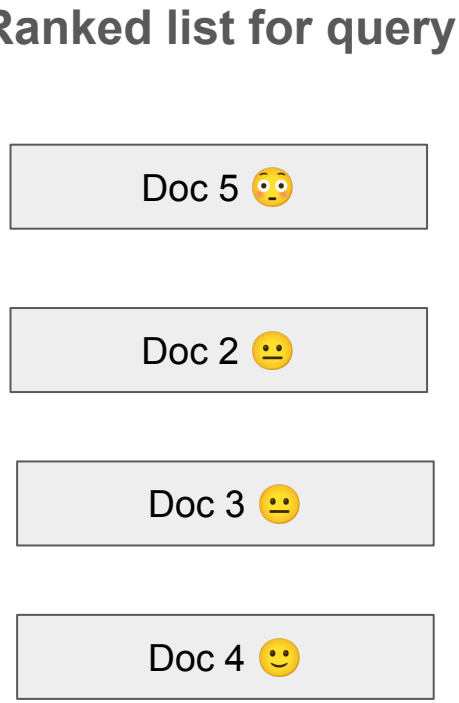
Judge query ⇔ document pairs

Binary judgment label:

- 🙂/😐

Graded judgment label:

- Very relevant, slightly relevant, irrelevant, my boss will fire me😳



Evaluating Information Retrieval Systems

Assess effectiveness and compare systems on IR relevancy datasets

Common IR datasets:

- Text REtrieval Conference (TREC)
  - Many collections spanning decades
  - Different domains and tasks
- MS Marco Passage/Document Ranking
  - Biggest training data (query, relevant doc)
  - Web search domain from Bing
- BEIR Benchmark (beir.ai)
  - Collection of IR collections
  - Many different domains and tasks
  - Zero-shot evaluation (no training data)

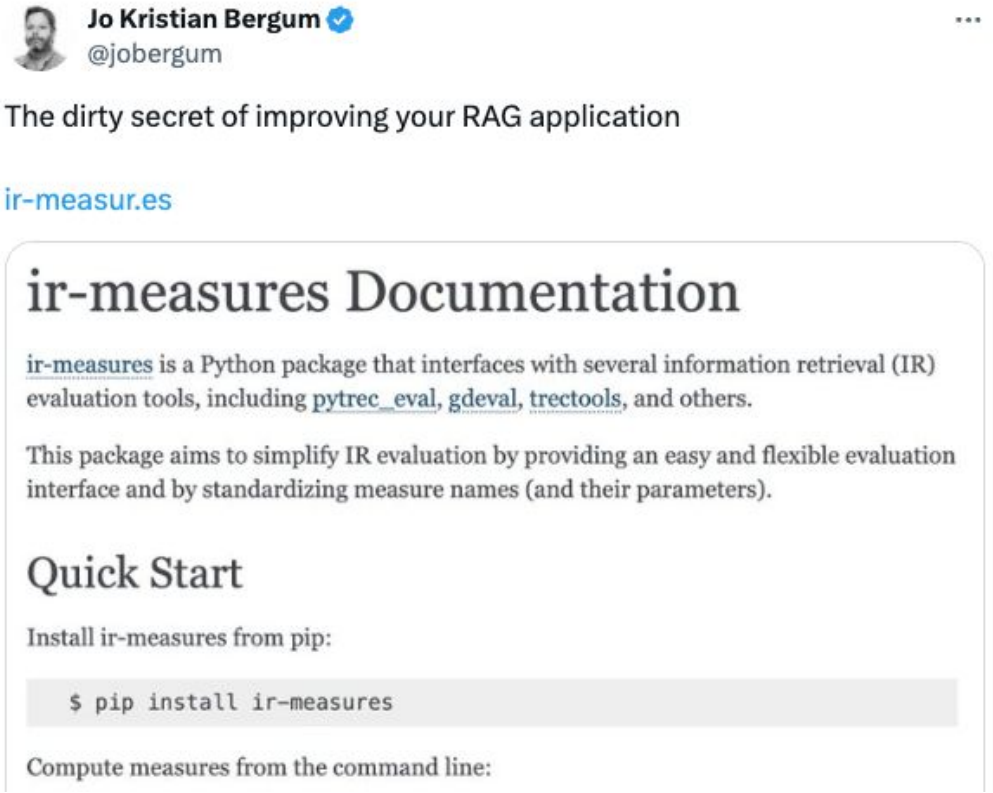
Ranking metrics

- **Recall@k (*All the relevant*)**
- **Precision@k (*Nothing but relevant)***
- ***nDCG@k (Graded, rank-aware)***
- ***Reciprocal Rank***
- **LGTM@10**
- **Industry:**
  - Engagement: click, dwell, add to chart
  - Revenue
  - Multi-objective ranking (*Not just optimizing relevance..*)
  - Note query distributions
    - Head (frequent) versus tail queries

Do better than LGTM@10

The dirty secret they have been hiding from you!

- Your data, your queries!
- Not some random IR dataset unrelated to your data.
- **Build your own relevance dataset**



Build your own golden relevance dataset

- Got real traffic? Sample those queries from head/torso
- No traffic? Ask a LLM to generate queries for your content
- **It doesn't need to be fancy** - A simple tsv file does the trick
- Preferably "static" collection (documents)

qid	docid	relevance label	comment
3 (how to ..)	4	2	
3 (where ..)	2	0	

In this example using graded relevance (0 irrelevant, 1 somehow relevant, 2 highly relevant)

Build your own relevance dataset

Golden dataset created by humans (you/domain experts)

Find prompt that correlates with golden dataset labels

Let the LLM evaluate more query and document pairs at scale

Eval, but also training/improving..

Large Language Models can Accurately Predict Searcher Preferences

Paul Thomas
Microsoft
Adelaide, Australia
pathom@microsoft.com

Seth Spielman
Microsoft
Boulder, USA
sethspielman@microsoft.com

Nick Craswell
Microsoft
Seattle, USA
nickcr@microsoft.com

Bhaskar Mitra
Microsoft Research
Montreal, Canada
bhaskar.mitra@microsoft.com

ABSTRACT

Much of the evaluation and tuning of a search system relies on relevance labels—annotations that say whether a document is useful for a given search and searcher. Ideally these come from real searchers, but it is hard to collect this data at scale, so typical experiments rely on third-party labellers who may or may not produce accurate annotations. Label quality is managed with ongoing auditing, training, and monitoring.

We discuss an alternative approach. We take careful feedback from real searchers and use this to select a large language model (LLM), and prompt, that agrees with this feedback; the LLM can then produce labels at scale. Our experiments show LLMs are as accurate as human labellers and as useful for finding the best systems and hardest queries. LLM performance varies with prompt features, but also varies unpredictably with simple paraphrases. This unpredictability reinforces the need for high-quality "gold" labels.

CCS CONCEPTS

• **Information systems** → *Test collections*; **Relevance assessment**.

KEYWORDS

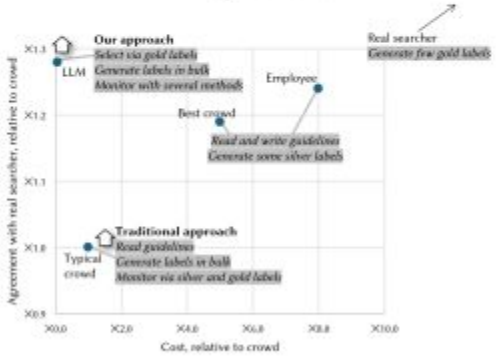


Figure 1: Labelling options discussed in this work, along with the cost and accuracy we see at Bing. A traditional approach uses gold and silver labels to improve crowd workers; we use gold labels to select LLMs and prompts.

Build your own relevance dataset

Sample prompt for relevance judgment

UMBRELA: UMbrela is the (Open-Source Reproduction of the Bing RELevance Assessor https://arxiv.org/abs/2406.06519 (June 10, 2024)

Easier than ever to build your own relevance dataset for your use case

LLM Judge also can free us from static golden collections, instead sample real user traffic.

```
Given a query and a passage, you must provide a score on an
integer scale of 0 to 3 with the following meanings:
0 = represent that the passage has nothing to do with the query,
1 = represents that the passage seems related to the query but
does not answer it,
2 = represents that the passage has some answer for the query,
but the answer may be a bit unclear, or hidden amongst extraneous
information and
3 = represents that the passage is dedicated to the query and
contains the exact answer.

Important Instruction: Assign category 1 if the passage is
somewhat related to the topic but not completely, category 2 if
passage presents something very important related to the entire
topic but also has some extra information and category 3 if the
passage only and entirely refers to the topic. If none of the
above satisfies give it category 0.

Query: {query}
Passage: {passage}

Split this problem into steps:
Consider the underlying intent of the search.
Measure how well the content matches a likely intent of the query
(M).
Measure how trustworthy the passage is (T).
Consider the aspects above and the relative importance of each,
and decide on a final score (O). Final score must be an integer
value only.
Do not provide any code in result. Provide each score in the
format of: ##final score: score without providing any reasoning.
```

LLM as relevance assessor (gpt4o)

Important to have a golden set to find correlation between LLM judge and human

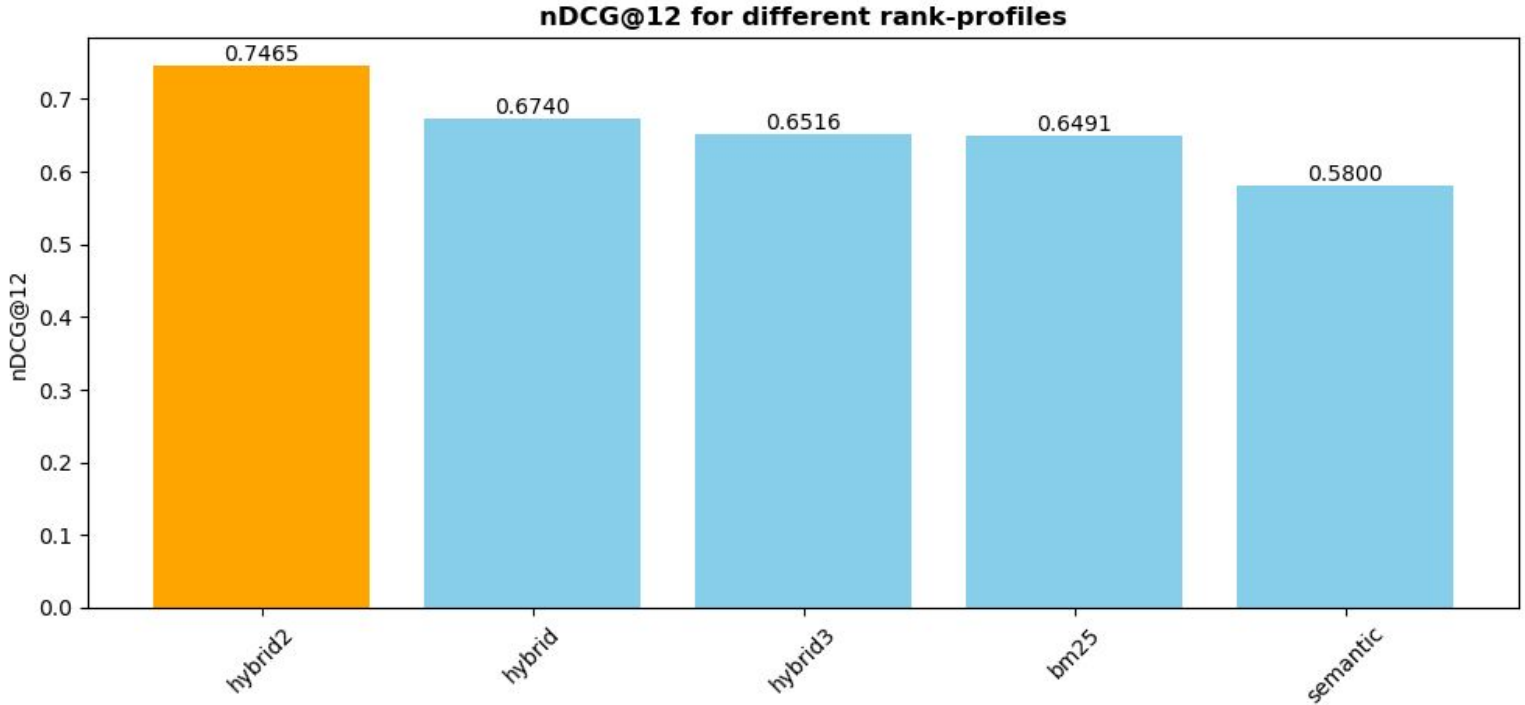
Golden set - 90 query, passage judgment pairs for Vespa documentation search

F1 Score: 0.89

12 True Negative	8 False Positive
8 False Negative	62 True Positive

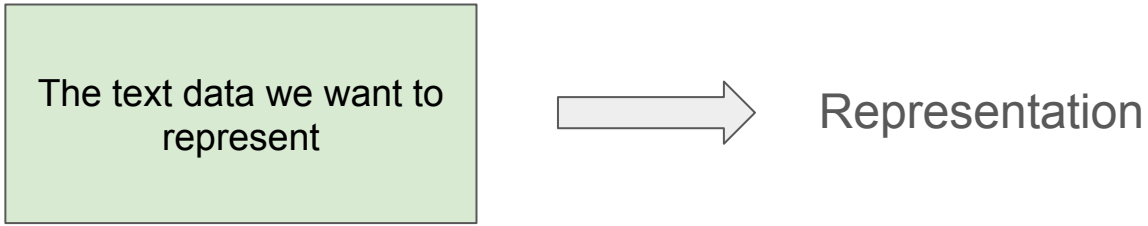
With your own eval dataset you can impress your CTO

From “*we changed title boost*” to “*we deployed a change that increases nDCG@12 with 30%*” https://github.com/vespa-cloud/vespa-documentation-search/tree/main/eval



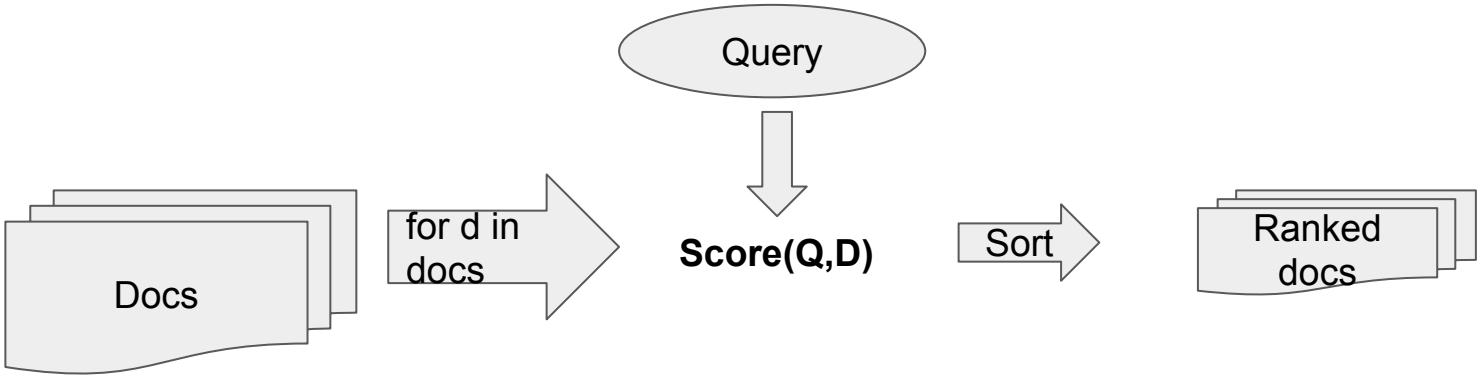
Representational approaches in IR

- Text representations
- Scoring functions



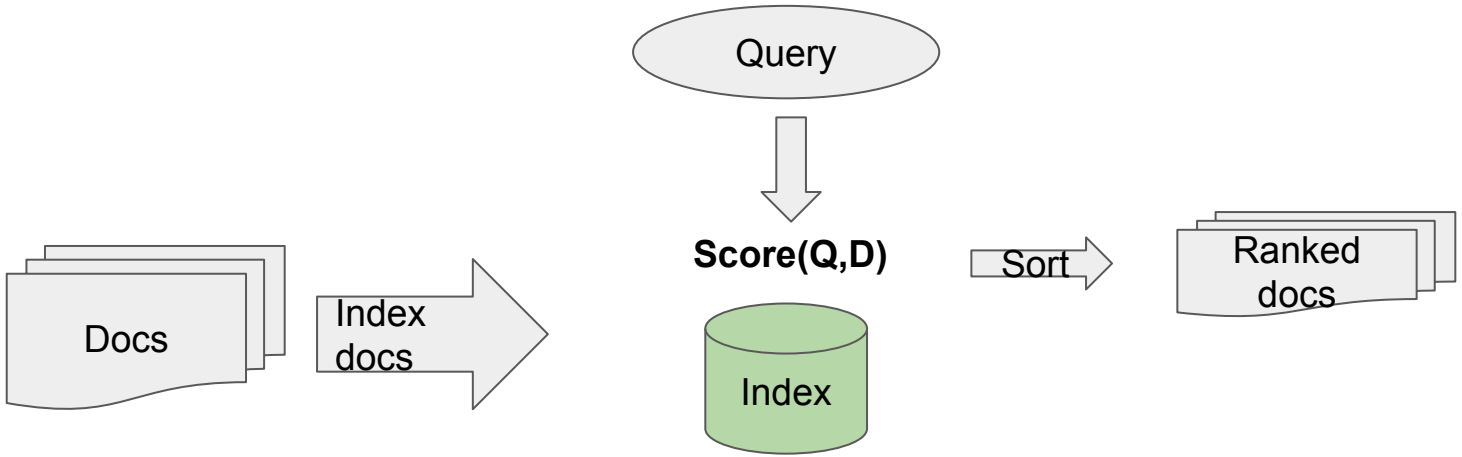
Motivation for representational IR

Avoid scoring all documents D in collection for a query Q



Motivation for representational IR

Avoid scoring all documents D in collection for a query Q



Text Representations

Into the implementation details

Accelerating retrieval over sparse representations

- Build inverted index data structures
- Search accelerated with top-k retrieval algorithms like WAND, MaxScore, BM-WAND and more
- Supervised (e.g splade) or unsupervised (e.g bm25, tf-idf)

Accelerating retrieval over dense representations

- Build vector index (IVF,PQ,HNSW, DiskANN++)
- Search accelerated (but approximate)
- Mainly supervised via transfer-learning (text embedding models)

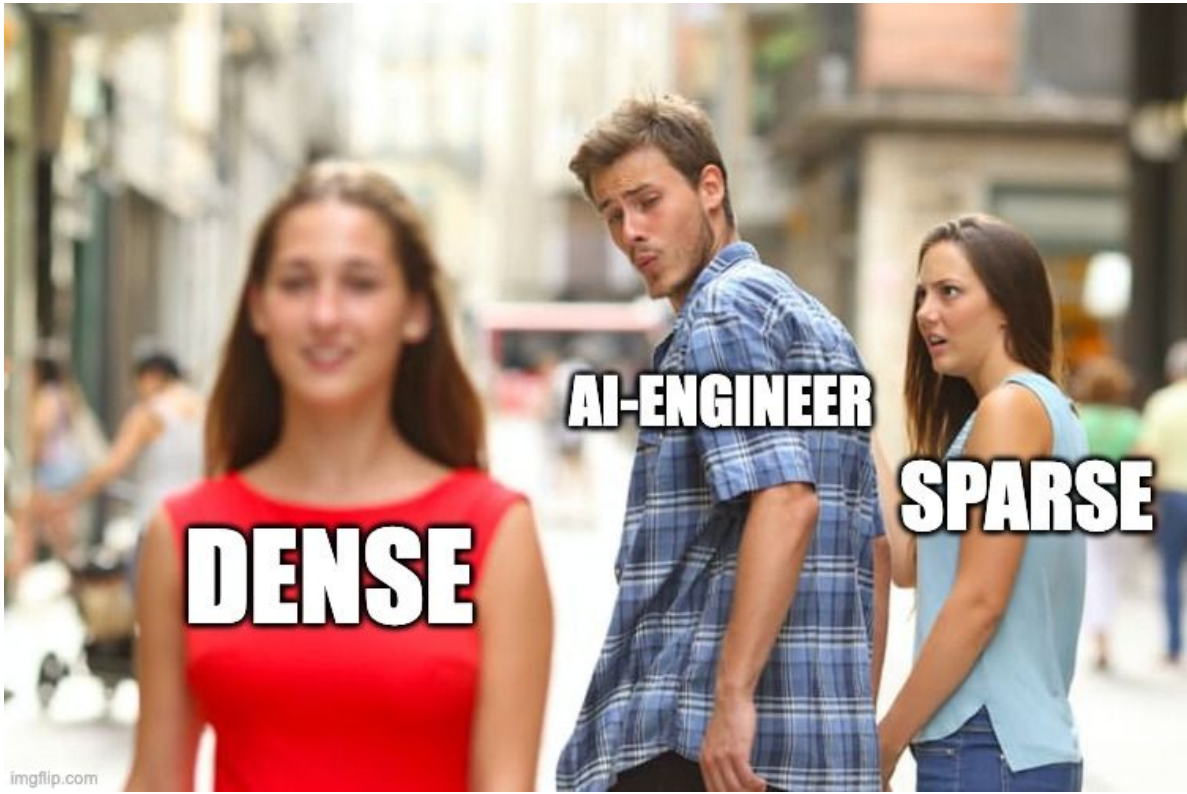
Text embeddings - Learned representations

Popularized by OpenAI text embeddings API

Represent queries and documents in a latent fixed d vector space

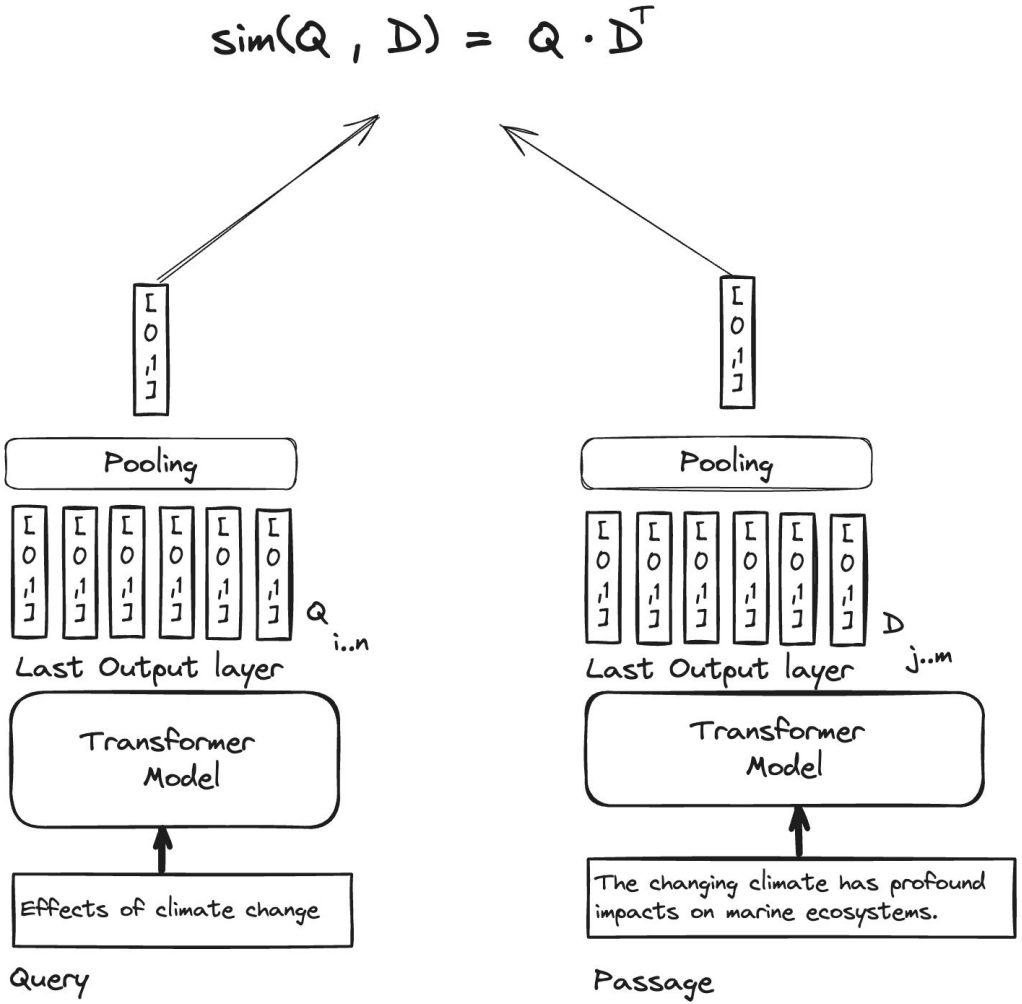
Score(q,d) = cosine similarity(q,d)/dot(q,d)

Accelerated retrieval via ANN (*but brute-force might be all you need*)



Challenges with embedding models

- Pooling dilutes long text representation
  - Require chunking
  - One doc - many chunks and vectors
  - Retrieve docs or chunks?
- Fixed vocabulary
  - GGUF Llama 3 2024 =>
  - ['g', '##gu', '##f', 'll', '##ama', '3', '202', '##4']
- Learned representation - transfer capabilities to your domain/data?



The (often?) missing baseline

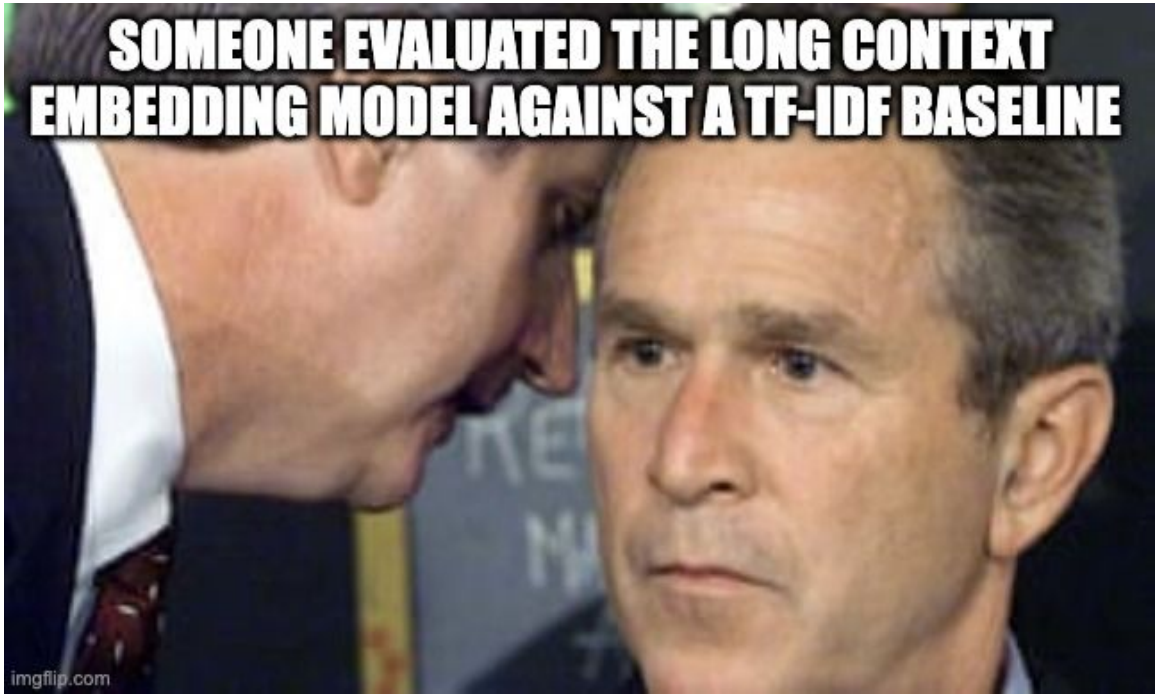
BM25 (Best Match 25) was designed for variable length texts. Statistical analysis of your data.

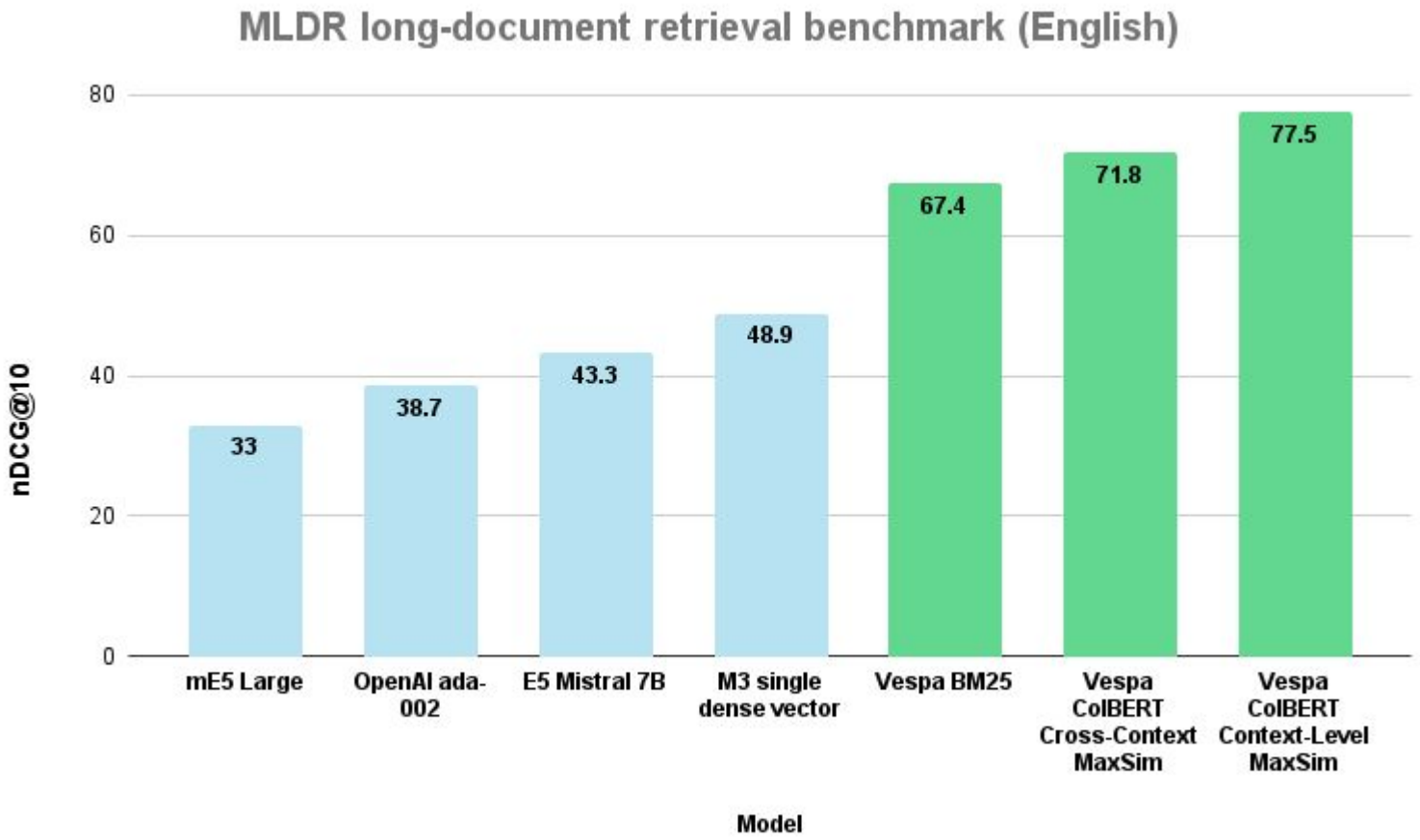
Cheap, small index footprint.

No embedding model inference required

Limited, but avoids spectacular failures

Also require tokenization and linguistic processing





From https://blog.vespa.ai/announcing-long-context-colbert-in-vespa/

Hybrid alternative

Combine best of both sparse and dense representations?

Overcome fixed vocab

Not a single silver bullet, but avoids common failure scenarios with single vector representations

All you need to know about chunking

Dense representations beyond 256 tokens are bad for high precision search

- Because the models haven’t been trained with longer sequences
- Longer texts drifts in topics

You need to chunk for meaningful vector representations for search

You don’t need to chunk a text into multiple rows and replicate metadata if using the right serving stack (vespa.ai)

Real world RAG

More than text similarity score

- Freshness
- Authority
- Quality
- Pagerank
- Revenue

Lots of features and ranking phases for retrieval at scale

GBDT is still king of tabular features

Report: 14,000+ Google Search Ranking Features Leaked

May 28, 2024 - 6:15 am 248 — by Barry Schwartz

Filed Under **Google Search Engine Optimization**

Rand Fishkin along with Mike King may have published one of the biggest data leaks outside of the **Department of Justice** reveal around Google Search and its internal ranking features and signals. The document was from an anonymous source (no longer anonymous, see below) but verified by Rand Fishkin and contains a ton of details on how Google Search reportedly works.

Summary

- IR is more than single-vector representation
- Build your own evals
- Don’t ignore the BM25 baseline
- Hybrid capabilities avoids the worst failure modes
- Long context single-vector embedding models underperforms
- Real-world search is more than text similarity

Resources

https://blog.vespa.ai/

Vespa RAG

https://search.vespa.ai/search?query=what%20is%20the%20benefit%20of%20colbert%20versus%20single%20vector%20models%3F&namespace=open-p,cloud-p,vespaapps-p,pyvespa-p

https://search.vespa.ai/search?query=what%20is%20multi-vector%20indexing%3F

Q & A

Hated it? Tweet me @jobergum