Data-Intensive Distributed Computing
CS 451/651 (Fall 2025)

UNIVERSITY OF WATERLOO

Text Processing II
(v1.00)

Week 10: November 6, 2025

Jimmy Lin
David R. Cheriton School of Computer Science
University of Waterloo

These slides are available at http://lintool.github.io/bigdata-2025f/

Key Questions

For sparse vector representations, how do we assign weights?
For sparse retrieval, how do we perform top-*k* retrieval efficiently?

For dense vector representations, how do we assign weights?
For dense retrieval, how do we perform top-*k* retrieval efficiently?

Do we need specialized databases for sparse and dense vectors?

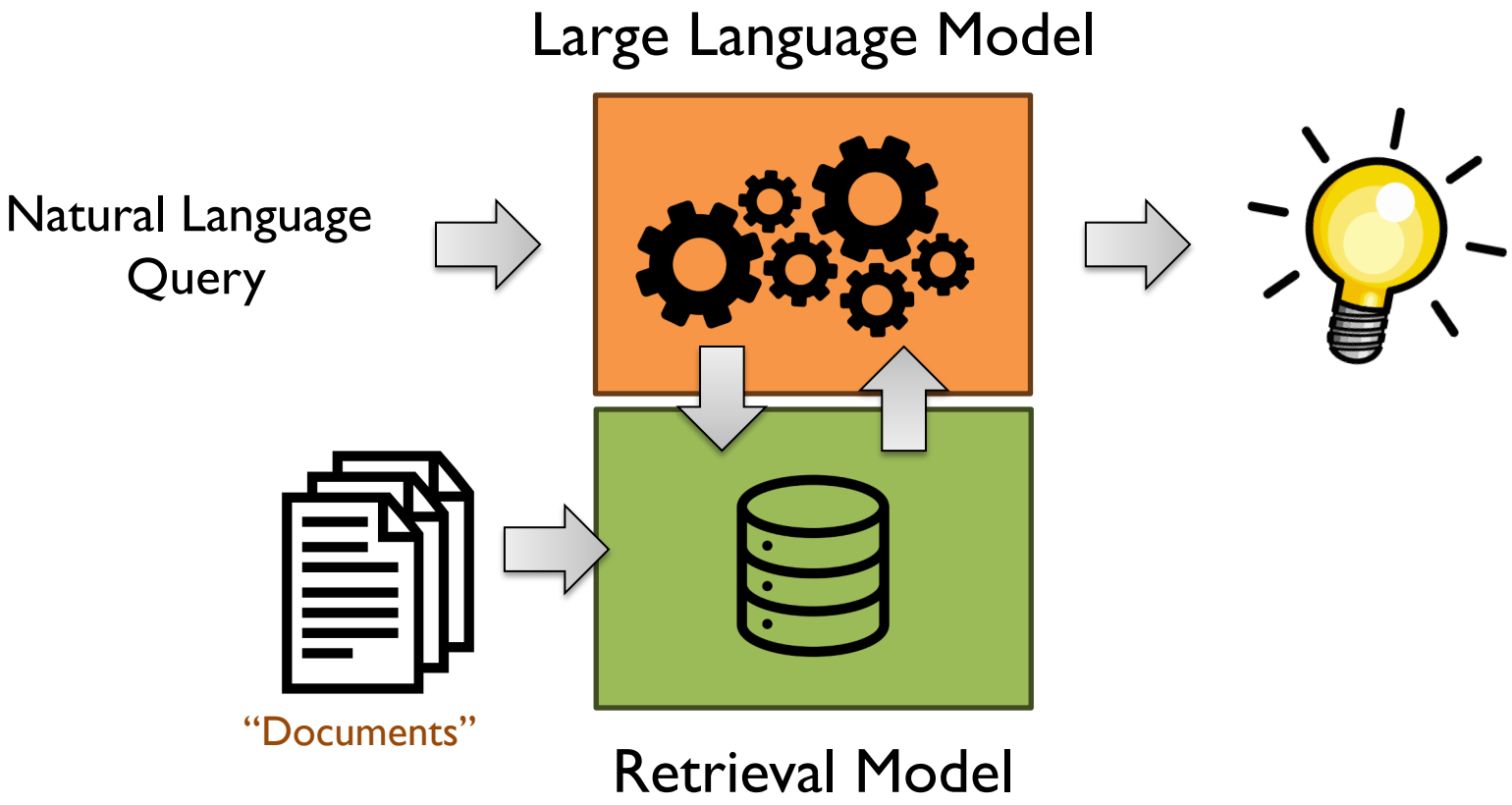
Can we also learn sparse representations from data?

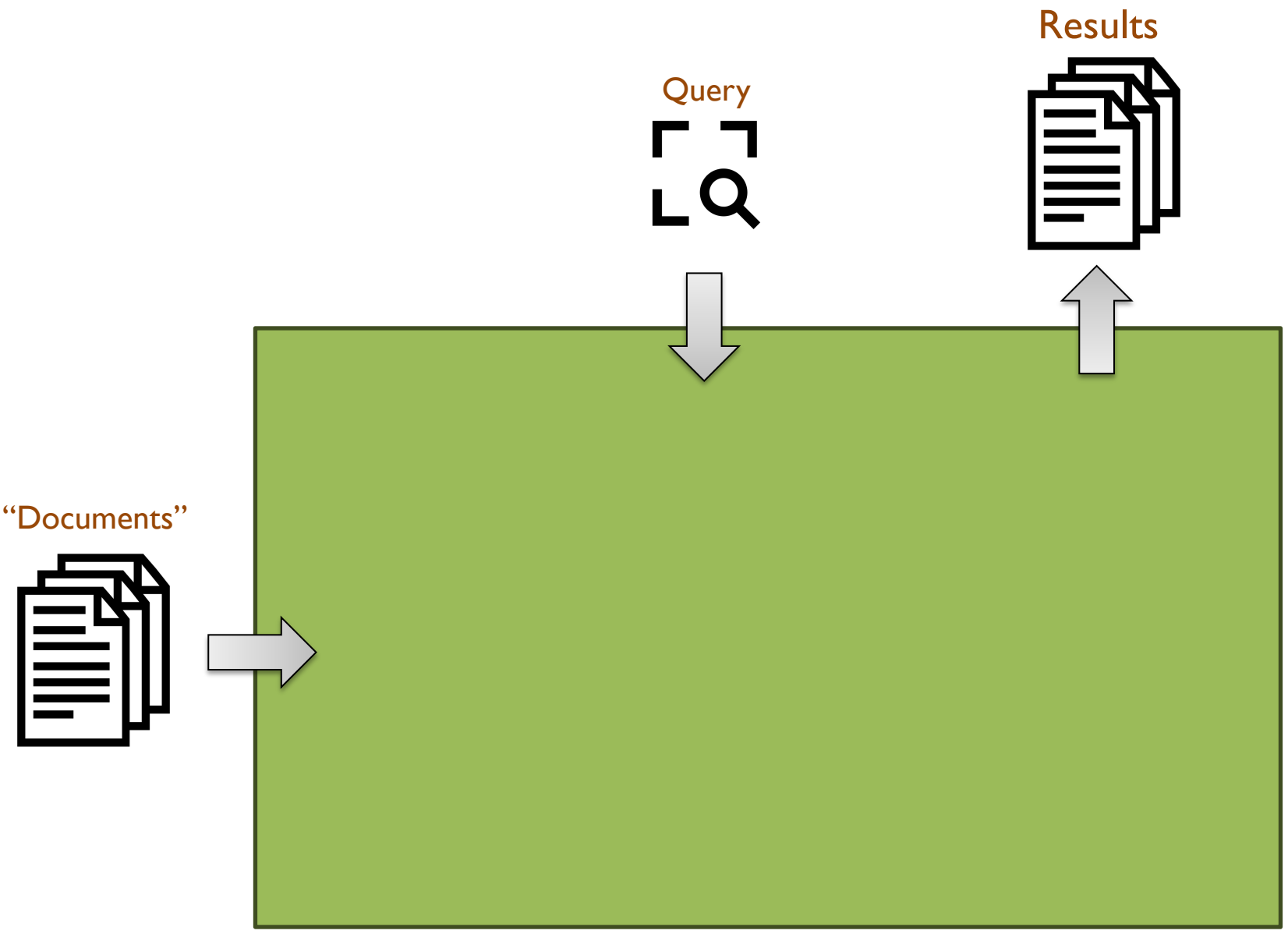
Why do we need rerankers?

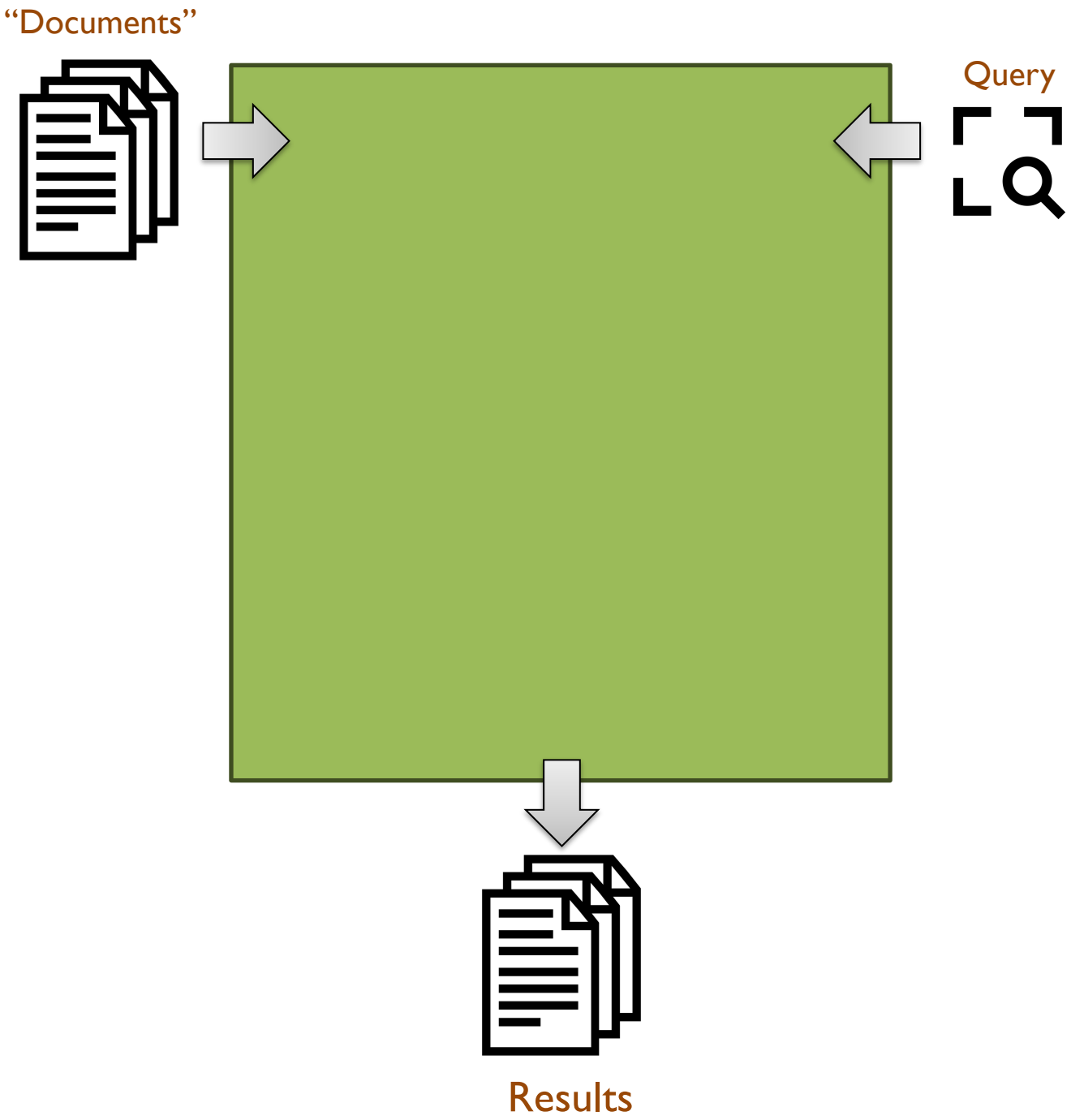
What's the relationship between RAG and LLM tool use?

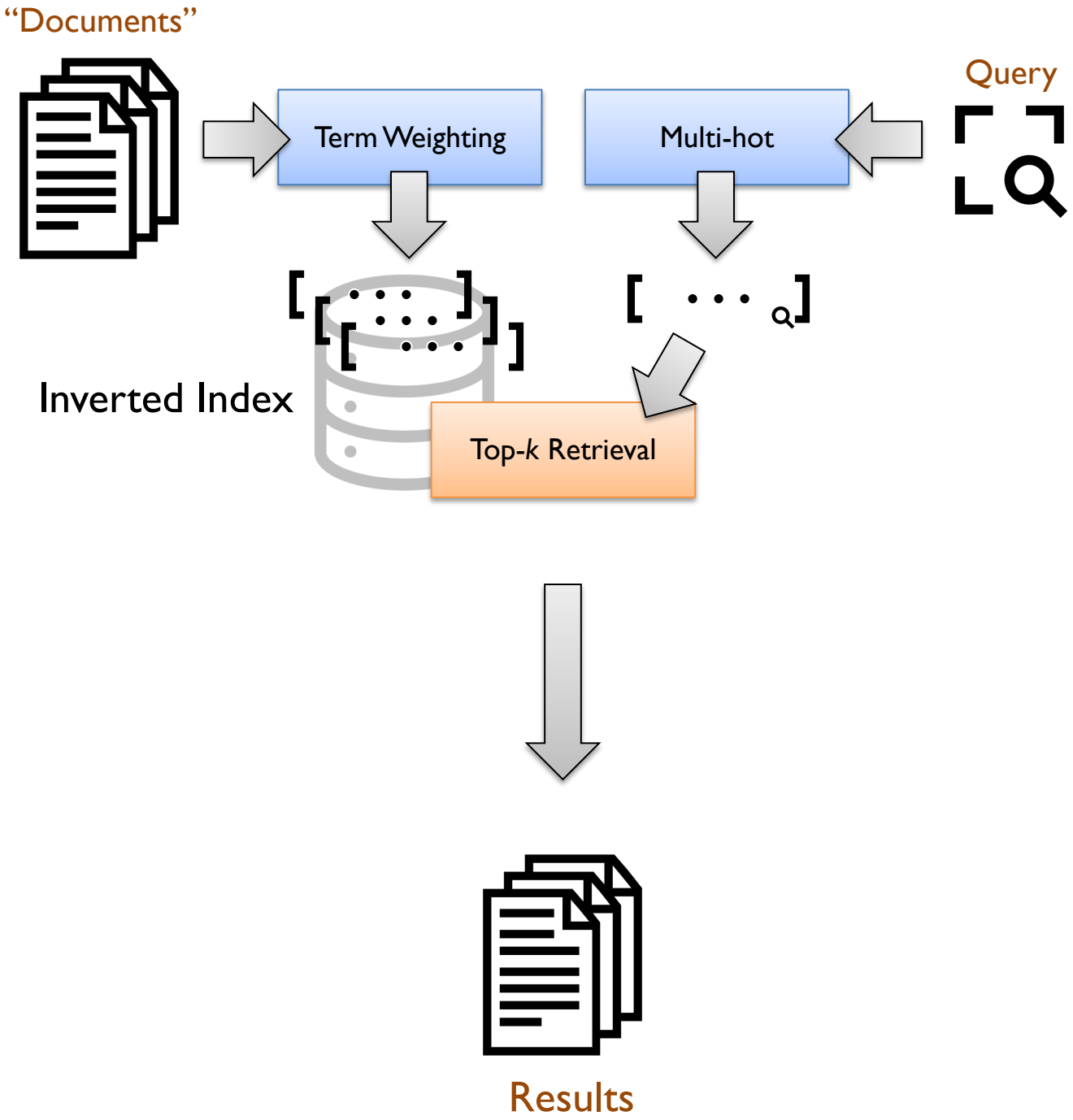
What challenges does MCP solve?

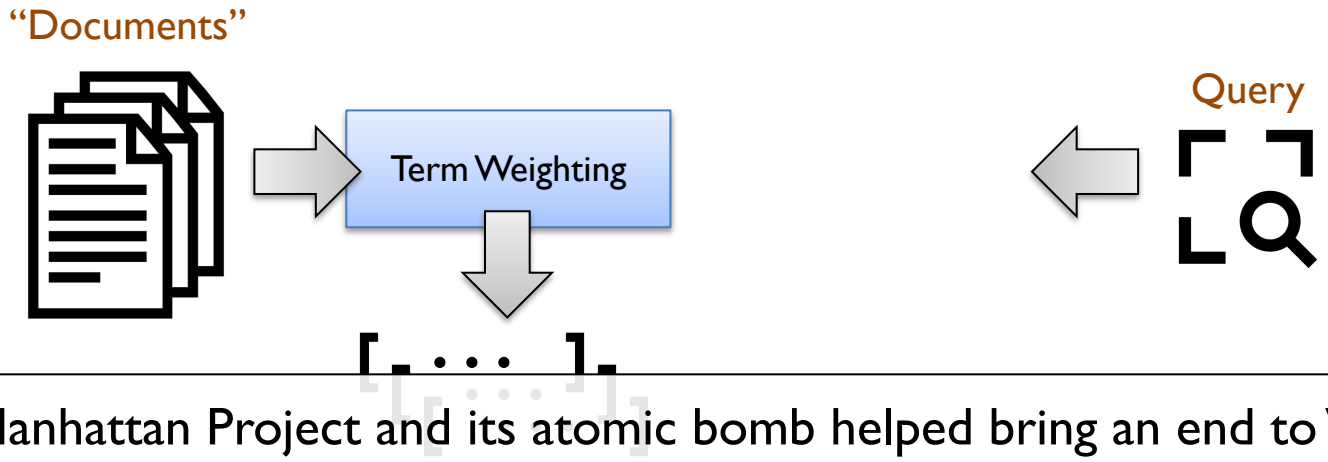
Retrieval-Augmented Generation




Results
Query
"Documents"


"Documents"
Query
Results


"Documents"
Query
Term Weighting
Multi-hot
Inverted Index
Top-k Retrieval
Results

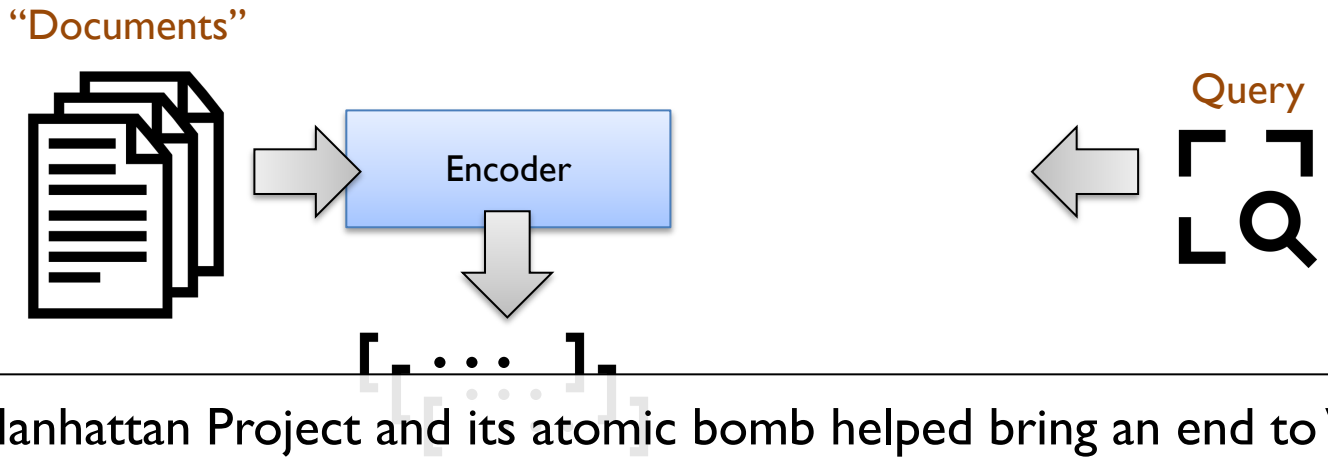


The Manhattan Project and its atomic bomb helped bring an end to World War II. Its legacy of peaceful uses of atomic energy continues to have an impact on history and science.

```
{'atom': 4.0140, 'bomb': 4.0704, 'bring': 2.7239, 'continu':
2.4331, 'end': 2.1559, 'energi': 2.5045, 'have': 1.0742, 'help':
1.8157, 'histori': 2.4213, 'ii': 3.0998, 'impact': 3.0304, 'it':
2.0473, 'legaci': 4.1335, 'manhattan': 4.1345, 'peac': 3.5205,
'project': 2.6442, 'scienc': 2.8700, 'us': 0.9967, 'war':
2.6454, 'world': 1.9974}
```

sparse vector

Results

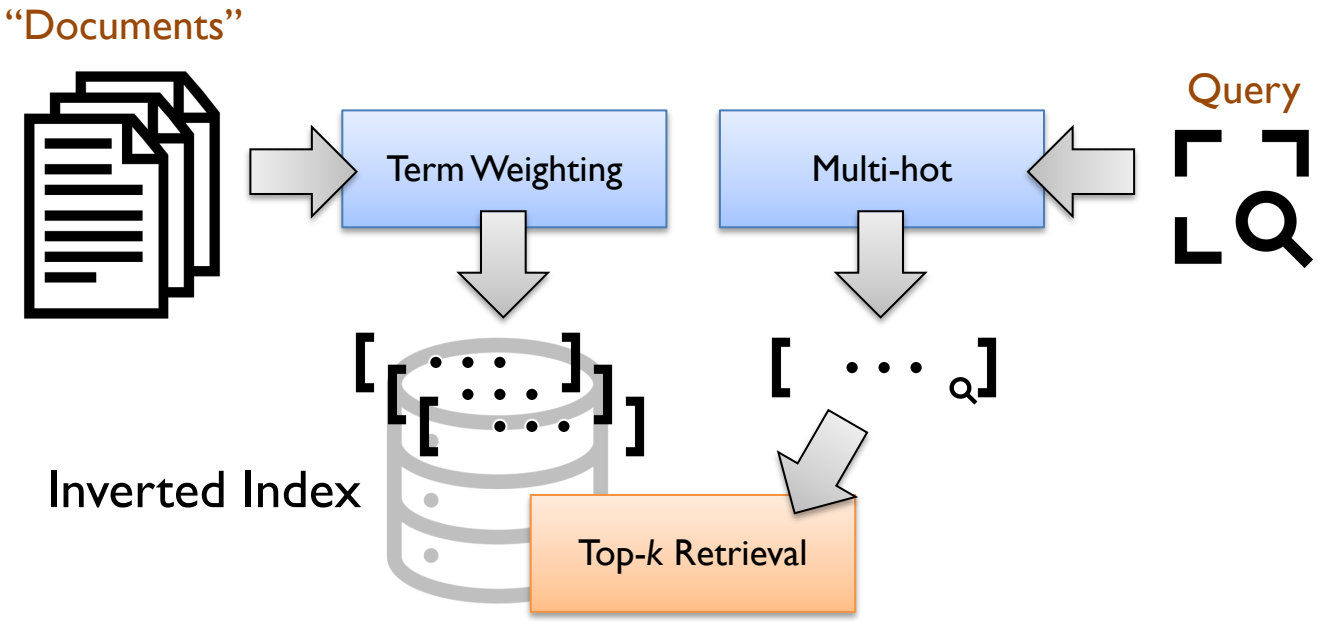


The Manhattan Project and its atomic bomb helped bring an end to World War II. Its legacy of peaceful uses of atomic energy continues to have an impact on history and science.

```
[0.099843978881836, 0.870057582855224 6, 0.520509719848633,
0.030491352081299, 0.723929882049560 5, 0.134523391723633,
0.433127403259277 3, 0.644286632537842, 0.645430564880371,
0.047342777252197 3, 0.070496082305908, 0.504533529281616,
0.815732955932617 2, 0.133575916290283, 0.997444868087768 6,
0.074254274368286 1, 0.155941247940063 5, 0.421395778656006,
0.014032363891602, 0.996794581413269...]
```

dense vector

Results



Challenge #1: How do we assign weights?

Challenge #2: How do we perform top-*k* retrieval efficiently?

Results

Challenge #1: Weight Assignment
Sparse vs. Dense Retrieval

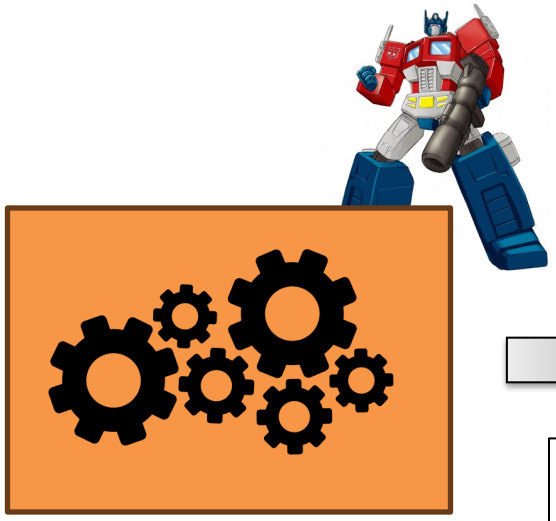
What's the basis of the vector?

How do we assign weights?	Sparse Bag of Words	Dense Latent Semantic Space
Supervised (via transformers)	?	✓
Unsupervised (heuristic)	BM25	

Learned Sparse Retrieval

The Manhattan Project and its atomic bomb helped bring an end to World War II. Its legacy of peaceful uses of atomic energy continues to have an impact on history and science.

Input



Model

Output

```
{'atom': 4.0140, 'bomb': 4.0704,
'bring': 2.7239, 'continu': 2.4331,
'end': 2.1559, 'energi': 2.5045, 'have':
1.0742, 'help': 1.8157, 'histori':
2.4213, 'ii': 3.0998, 'impact': 3.0304,
'it': 2.0473, 'legaci': 4.1335,
'manhattan': 4.1345, 'peac': 3.5205,
'project': 2.6442, 'scienc': 2.8700,
'us': 0.9967, 'war': 2.6454, 'world':
1.9974}
```

Like BM25, but better weights!

Example: Lassance et al. SPLADE-v3: New baselines for SPLADE. *arXiv:2403.06789*.

Learned Sparse Retrieval

Lexical vector representations

Term weights are learned from data

Incorporates term expansion in encoding

vs. dense vector retrieval

Similar retrieval quality

Don't need specialized vector indexes

vs. BM25

Better retrieval quality

Can also use inverted indexes (via "fake words" trick)

Best of both worlds?

Example: Lassance et al. SPLADE-v3: New baselines for SPLADE. *arXiv:2403.06789*.

Challenge #1: Weight Assignment
Sparse vs. Dense Retrieval

How do we assign weights?	What's the basis of the vector? Sparse Bag of Words	Dense Latent Semantic Space
Supervised (via transformers)	✓	✓
Unsupervised (heuristic)	BM25	

Challenge #2: Efficient Top-*k* Retrieval

Do you need both?

Sparse vectors
(BM25 or learned)

"text databases"

Dense vectors

vector databases

Challenge #2: Efficient Top-*k* Retrieval

Do you need both?

Yes, hybrid fusion – combining results of both – often outperforms individual results

Min/Max normalization + Score averaging

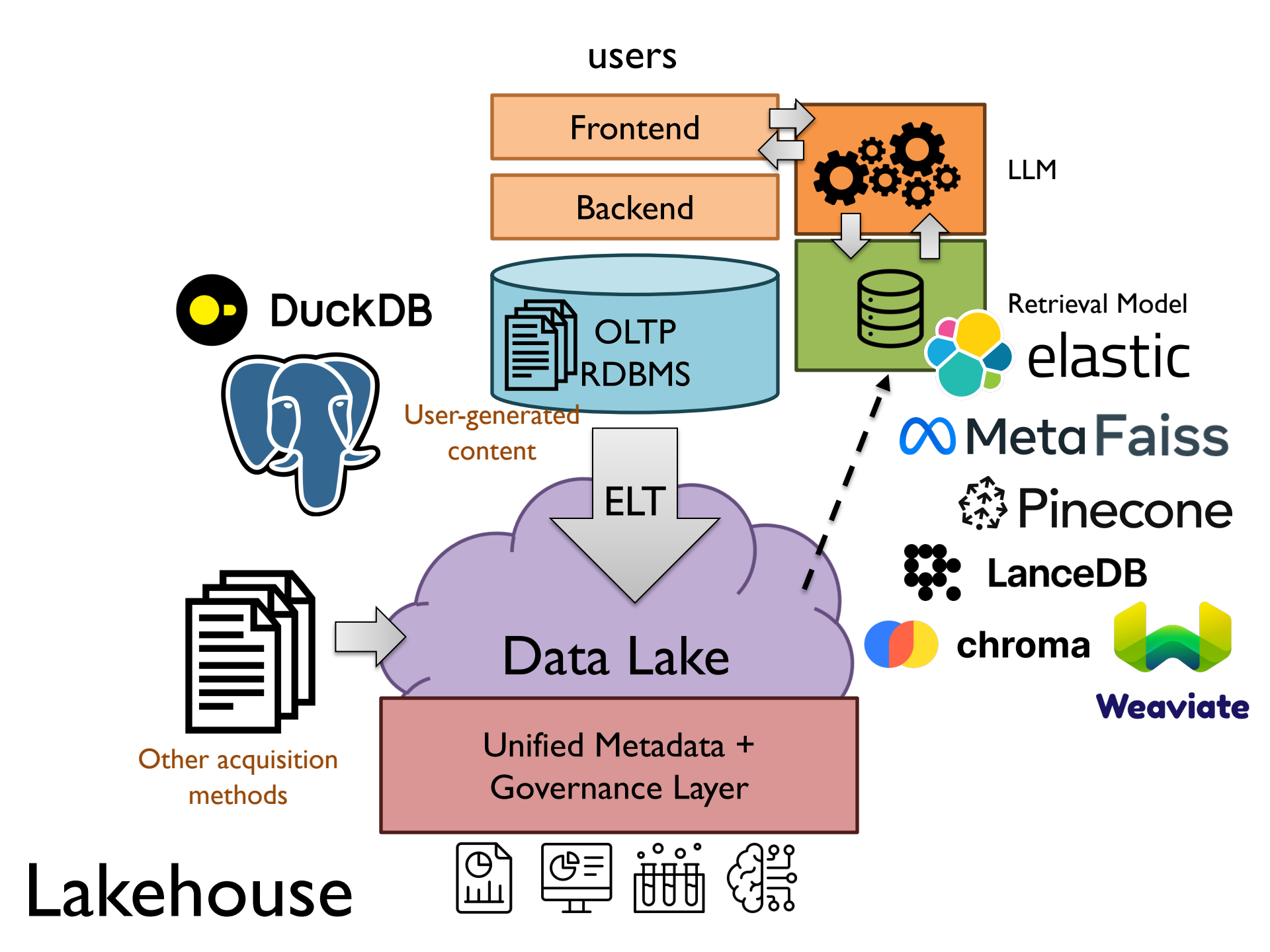
Reciprocal Rank Fusion

RRF simply sorts the documents according to a naive scoring formula. Given a set D of documents to be ranked and a set of rankings R, each a permutation on $1..|D|$, we compute

$$RRFscore(d \in D) = \sum_{r \in R} \frac{1}{k + r(d)},$$

where $k = 60$ was fixed during a pilot investigation and not altered during subsequent validation. Our intuition in

Cormack et al. Reciprocal Rank Fusion outperforms Condorcet and individual Rank Learning Methods. *SIGIR 2009*.

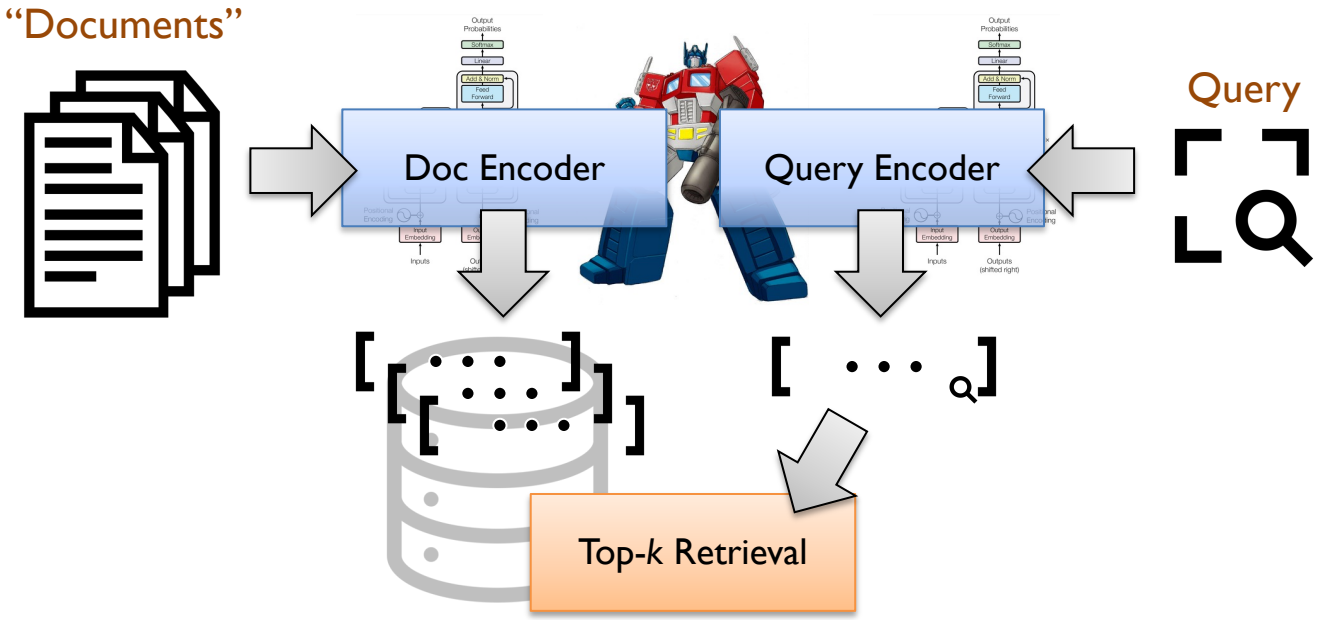

users
Frontend
Backend
LLM
DuckDB
OLTP
RDBMS
Retrieval Model
elastic
User-generated
content
Meta Faiss
ELT
Pinecone
LanceDB
chroma
Weaviate
Data Lake
Unified Metadata +
Governance Layer
Other acquisition
methods
Lakehouse

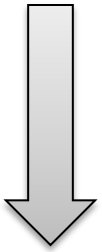
Sparse vs. Dense Retrieval

What's the basis of the vector?

How do we assign weights?	Sparse Bag of Words	Dense Latent Semantic Space
	Use data if you have it, *synthesize* data if you don't!	
Supervised (via transformers)	✓	✓
Unsupervised (heuristic)	BM25	?

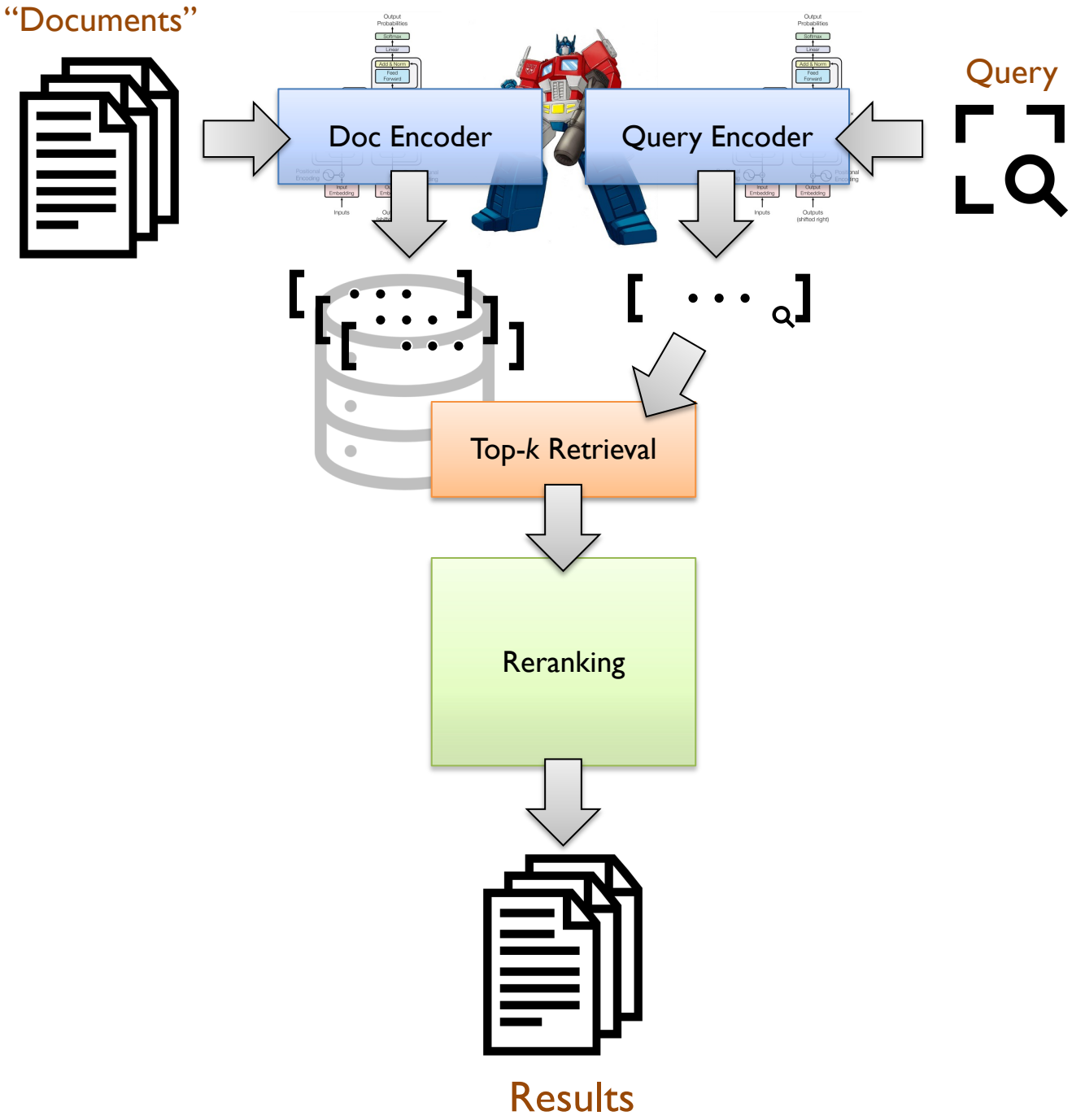
Work does exist…
but nothing recent

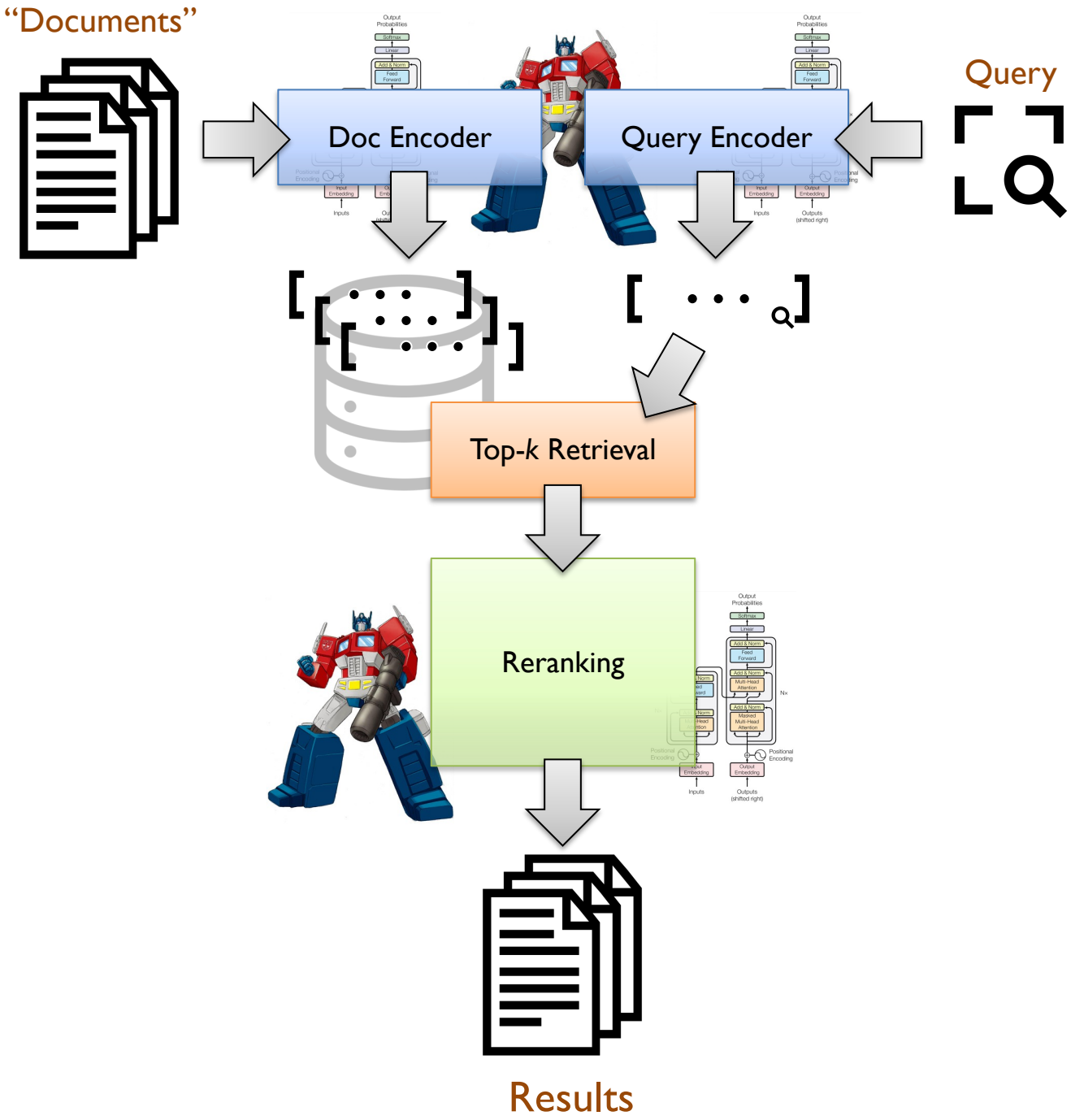




Results

But wait, there's more!

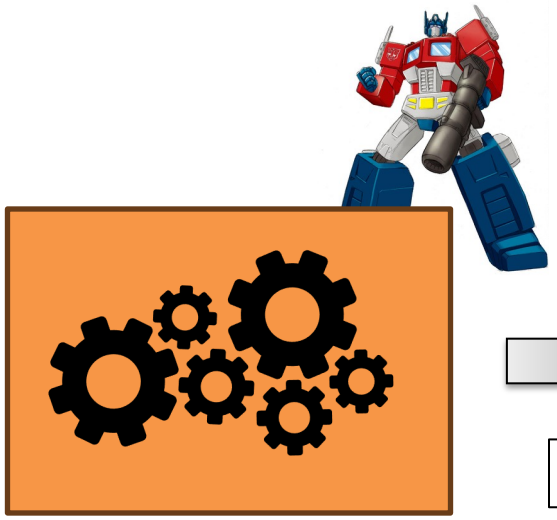

"Documents"
Doc Encoder
Query Encoder
Query
Top-k Retrieval
Reranking
Results


"Documents"
Doc Encoder
Query Encoder
Query
Top-k Retrieval
Reranking
Results

Pointwise

For the following query, is the document relevant?

Q: …
D: …



Input → **Model** → **Output**

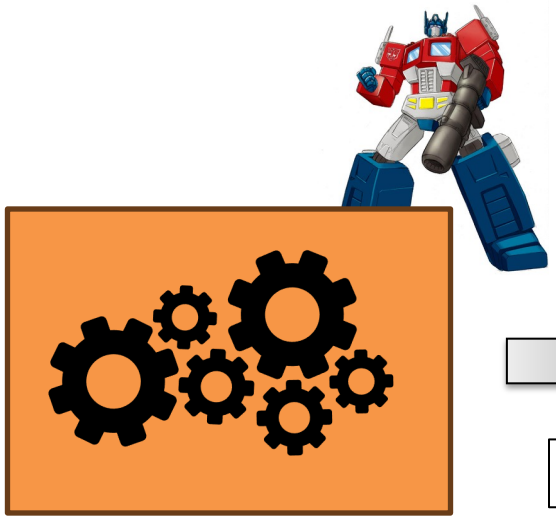
0.923

Pairwise

For the following query, which document is more relevant?

Q: …
D1: …
D2: …

Input



Model

Output

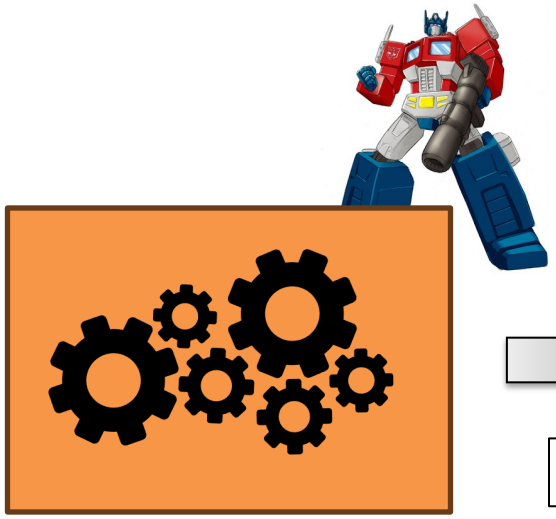
D1

Listwise

For the following query, rank the documents in order of relevance.

Q: …
D1: …
D2: …
D3: …
D4: …

Input



Model

Output

```
D3 > D2 > D4 > D1
```

Remember this?

The Task

Given: $D = \{(\mathrm{x}_i, y_i)\}_i^n$

label

feature vector

$$\mathrm{x}_i = [x_1, x_2, x_3, \ldots, x_d]$$
$$y \in \{0, 1\}$$

Induce: $f : X \rightarrow Y$

Such that loss is minimized

$$\frac{1}{n} \sum_{i=0}^{n} \ell(f(\mathrm{x}_i), y_i)$$

loss function

Typically, we consider functions of a parametric form:

$$\arg\min_{\theta} \frac{1}{n} \sum_{i=0}^{n} \ell(f(x_i; \theta), y_i)$$

model parameters

And this?

Challenge #1: Term Weights

Q: Where do these dense vectors (= embeddings) come from?

A: They're learned from data!

Let $\mathcal{D} = \{\langle q_i, p_i^+, p_{i,1}^-, \cdots, p_{i,n}^- \rangle\}_{i=1}^m$ be the training data that consists of m instances. Each instance contains one question q_i and one relevant (positive) passage p_i^+, along with n irrelevant (negative) passages $p_{i,j}^-$. We optimize the loss function as the negative log likelihood of the positive passage:

$$L(q_i, p_i^+, p_{i,1}^-, \cdots, p_{i,n}^-) = -\log \frac{e^{\text{sim}(q_i, p_i^+)}}{e^{\text{sim}(q_i, p_i^+)} + \sum_{j=1}^n e^{\text{sim}(q_i, p_{i,j}^-)}}. \quad (2)$$

Intuition: you have many queries with relevant and non-relevant examples

inner product of (queries, relevant examples) → make this high

inner product of (queries, non-relevant examples) → make this low

Karpukhin et al. Dense Passage Retrieval for Open-Domain Question Answering. *EMNLP 2020.*

Rerankers are trained in similar ways…

Why rerank?

In general, there's a tradeoff between speed and quality

Rerankers are typically slower than retrievers

Tradeoff between quality and speed due to amount of "interaction"

Retrieve → rerank: balance quality and speed

(lower quality, faster) → (higher quality, slower)

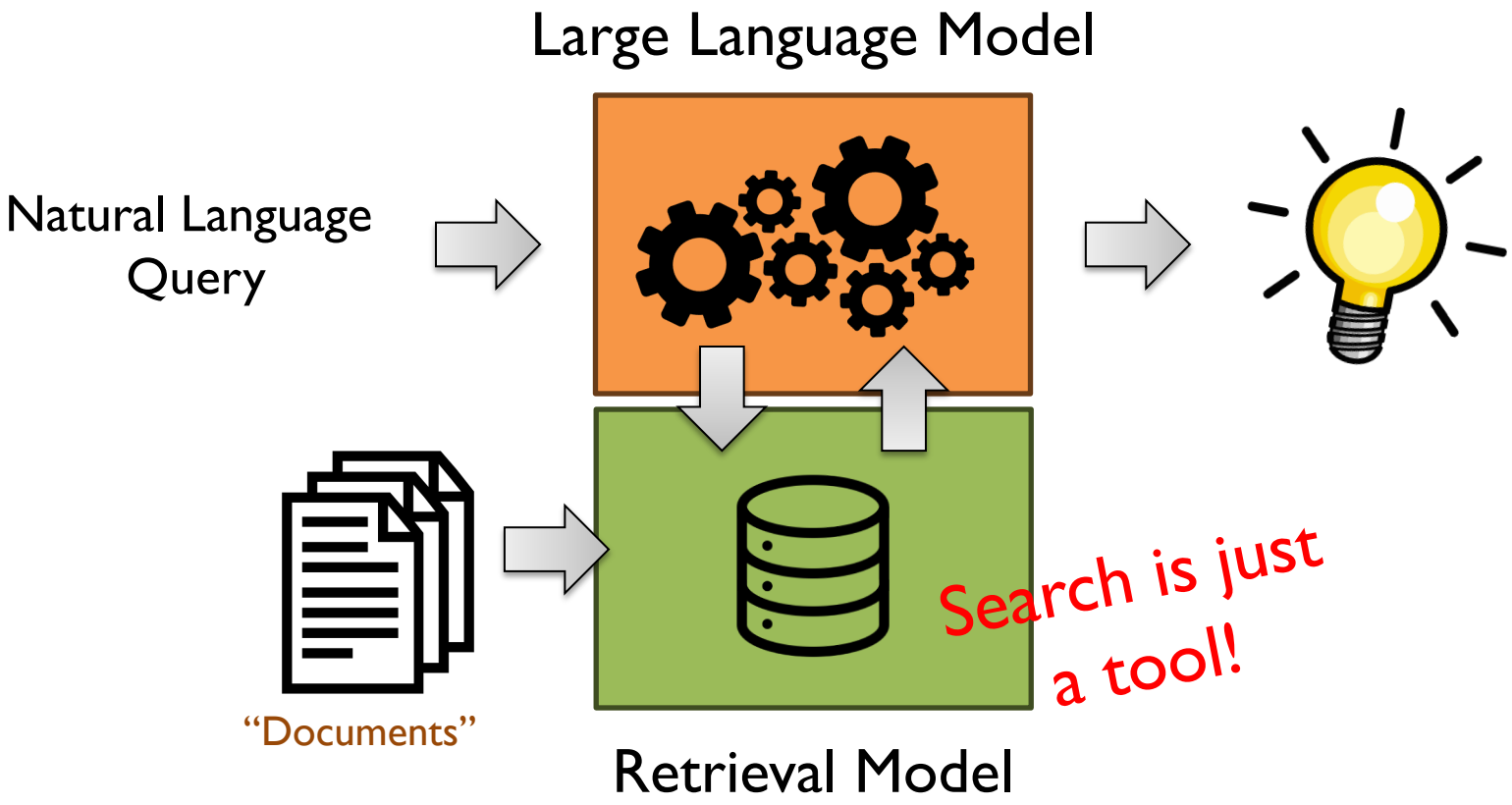
(more, but faster) → (fewer, but slower)

Retrieve-Rank Architecture

Beauty of this approach is that you can embed anything!

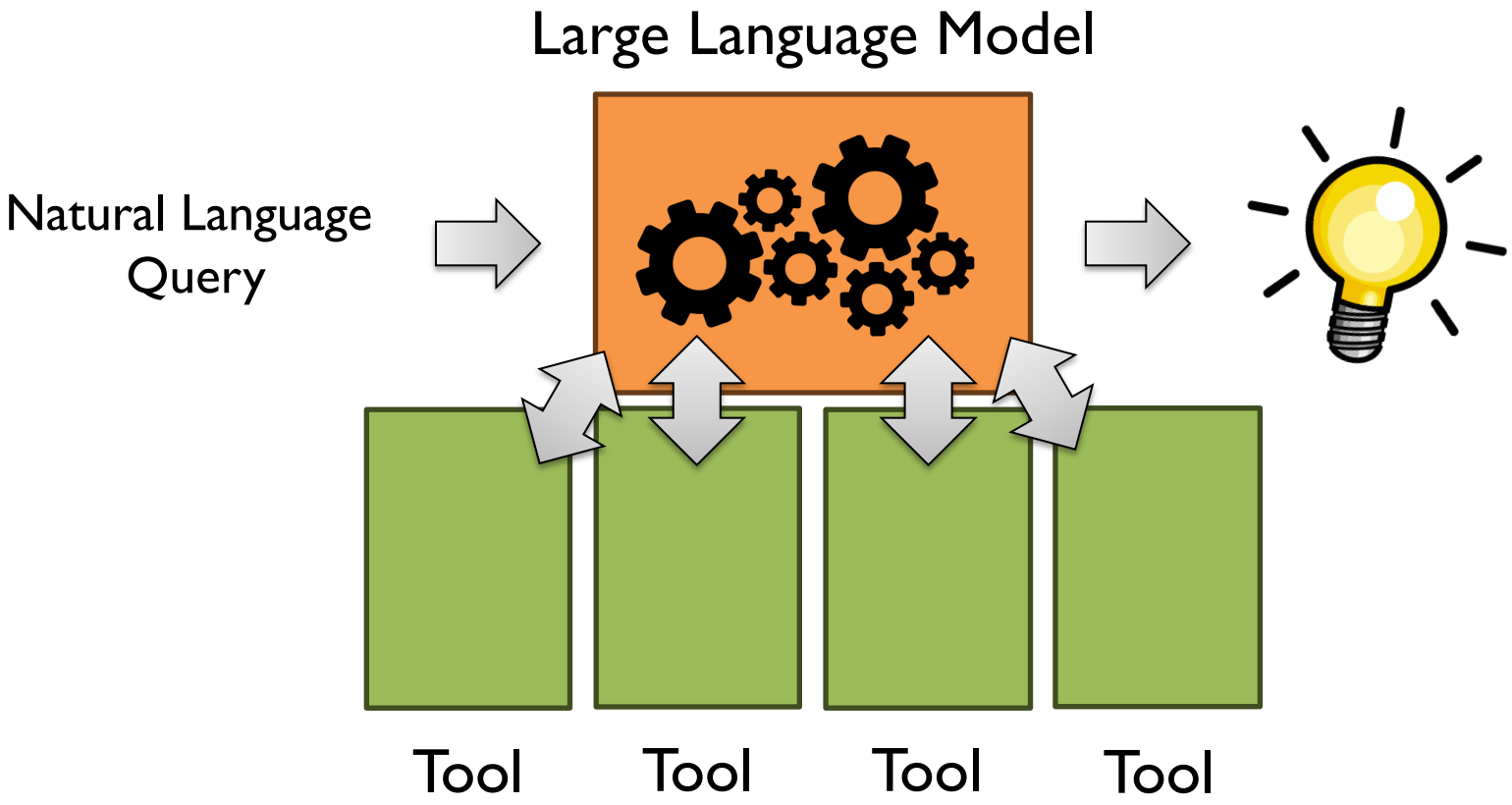
Beauty of this approach is the applicability to many applications!

Retrieval-Augmented Generation



LLM Tool Use

LLMs can use + combine
results from different tools!



LLM Tool Use

Challenge #1: What happens as the number of tools increases?

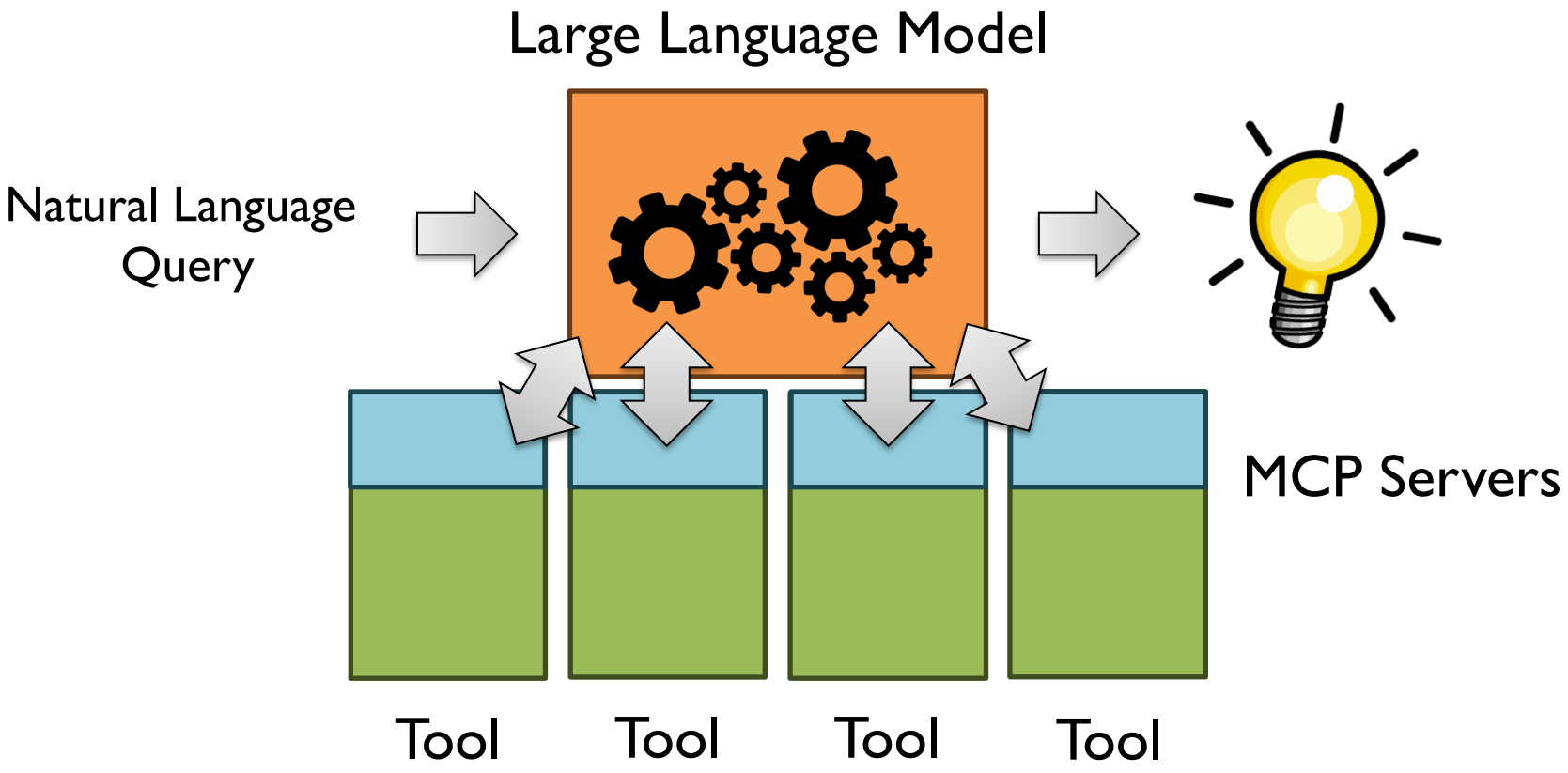
Cost of "wiring up" LLMs with REST APIs

Challenge #2: What happens if we want to change the LLM?

Cost of "wiring up" *n* tools with *m* LLMs

MCP (Model Context Protocol)

An open-source standard for connecting AI applications to external systems




富嶽三十六景
神奈川沖
浪裏