Retrieval-Augmented Generation (RAG)

Michael T. Goodrich
University of California, Irvine

Slides from Graham Neubig, Umar Jamil, and Haofen Wang

Standard Prompting

- Combine a prompt template together with an input

Please answer this question:

I think Vin Diesel has been a voice actor for several characters in TV series, do you know what their names are?

Problems

- *Accuracy issues:*
  - **Knowledge cutoffs:** parameters are usually only updated to a particular time
  - **Private data:** data stored in private text or data repositories not suitable for training
  - **Learning failures:** even for data that the model was trained on, it might not be sufficient to get the right answer
- *Verifiability issues:* It is hard to tell if the answer is correct

I asked Google Gemini:

How do WWW links get incorporated with your LLM?

That's a great question that gets to the heart of how modern AI assistants function! I don't "absorb" the entire internet into my brain. Instead, I use a process called **Retrieval-Augmented Generation (RAG)** in conjunction with my Google Search tool.

This allows me to leverage the vast knowledge of my original training data while simultaneously accessing real-time, external information.

Here is a breakdown of the three-step process:

1. Tool Call and Retrieval (The "R" in RAG)

When you ask a question that requires current or specific factual information (like "Who won the World Series this year?" or a niche scientific term), my internal logic determines that a web search is necessary.

- **The Model's Decision:** I analyze your prompt and decide to activate the `google:search` tool.
- **Search Execution:** The tool instantly executes one or more search queries (like a conventional Google search).
- **Retrieval:** The tool returns a list of highly **ranked search results** (snippets, titles, and URLs) from Google's constantly updated index. These snippets contain the most relevant, authoritative, and up-to-date facts related to your query.

2. Context Augmentation (The Integration)

The key step is how this external information is fed into my language model architecture.

- **Augmented Prompt:** The system takes your original question and **augments** it by adding the text from the top search snippets. This combined text—your query plus the fresh facts—is packaged into a single, extended input, or **prompt**.
- **The Role of Context Window:** The prompt is sent to my core language model. I use my large **context window** (the memory capacity for input text) to hold both your original intent and the retrieved facts simultaneously.

This process essentially gives me a temporary, real-time "note" of the facts I need to answer the question accurately, **grounding** my response in verified data.

3. Generation and Citation (The "G" in RAG)

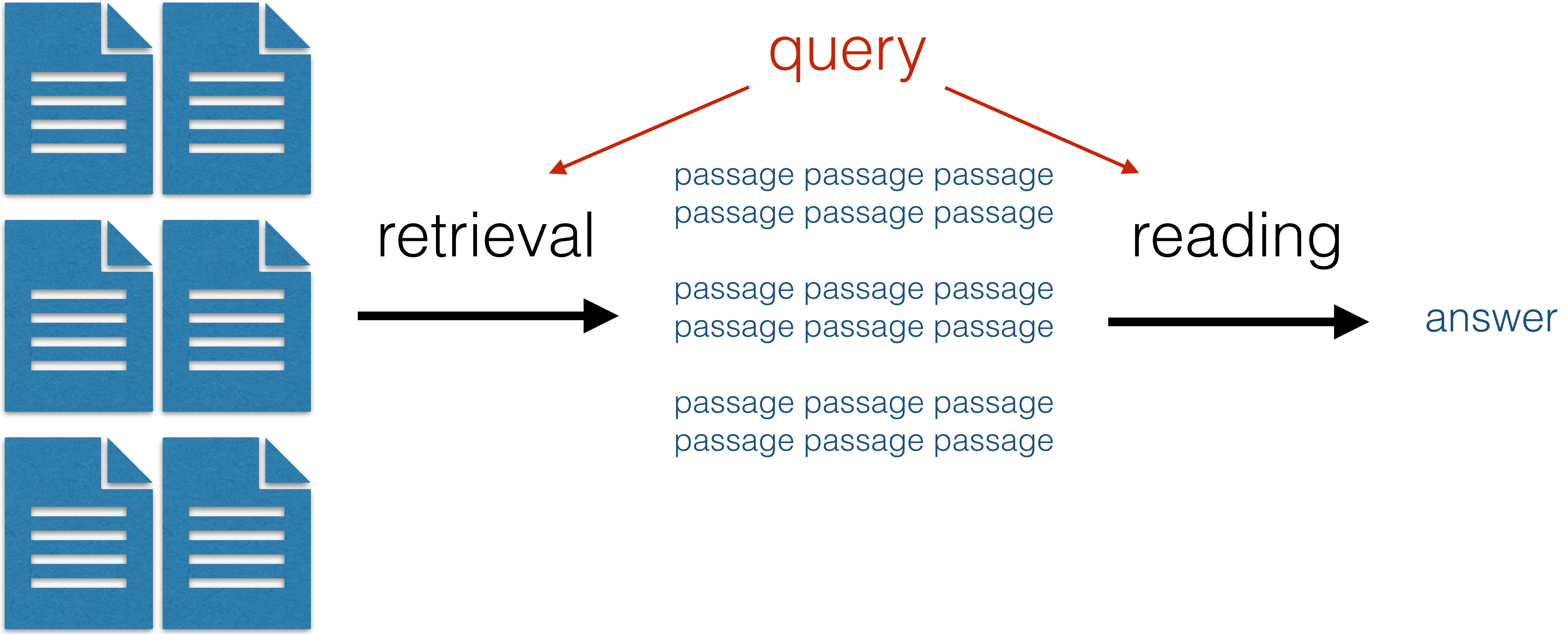
Finally, my core language model synthesizes the answer.

- **Generation:** I use my deep language understanding to read the augmented prompt (containing the retrieved facts) and generate a coherent, human-readable response that directly answers your question.
- **Fact-Checking:** Because I am forced to rely on the retrieved snippets, this process drastically **reduces the chance of "hallucinations"** (making up facts).
- **Citation:** The system tracks which specific search result provided the information used in the final response. This allows me to automatically provide the **source references (links)** in the output, building trust and allowing you to verify the claims.

This RAG architecture is what transforms an LLM from a static knowledge base (only knowing what it was trained on) into a **dynamic assistant** that can provide current and verifiable information.

Retrieval-augmented Generation
(Chen et al. 2017)

- **Retrieve** relevant passages efficiently
- **Read** the passages to answer the query

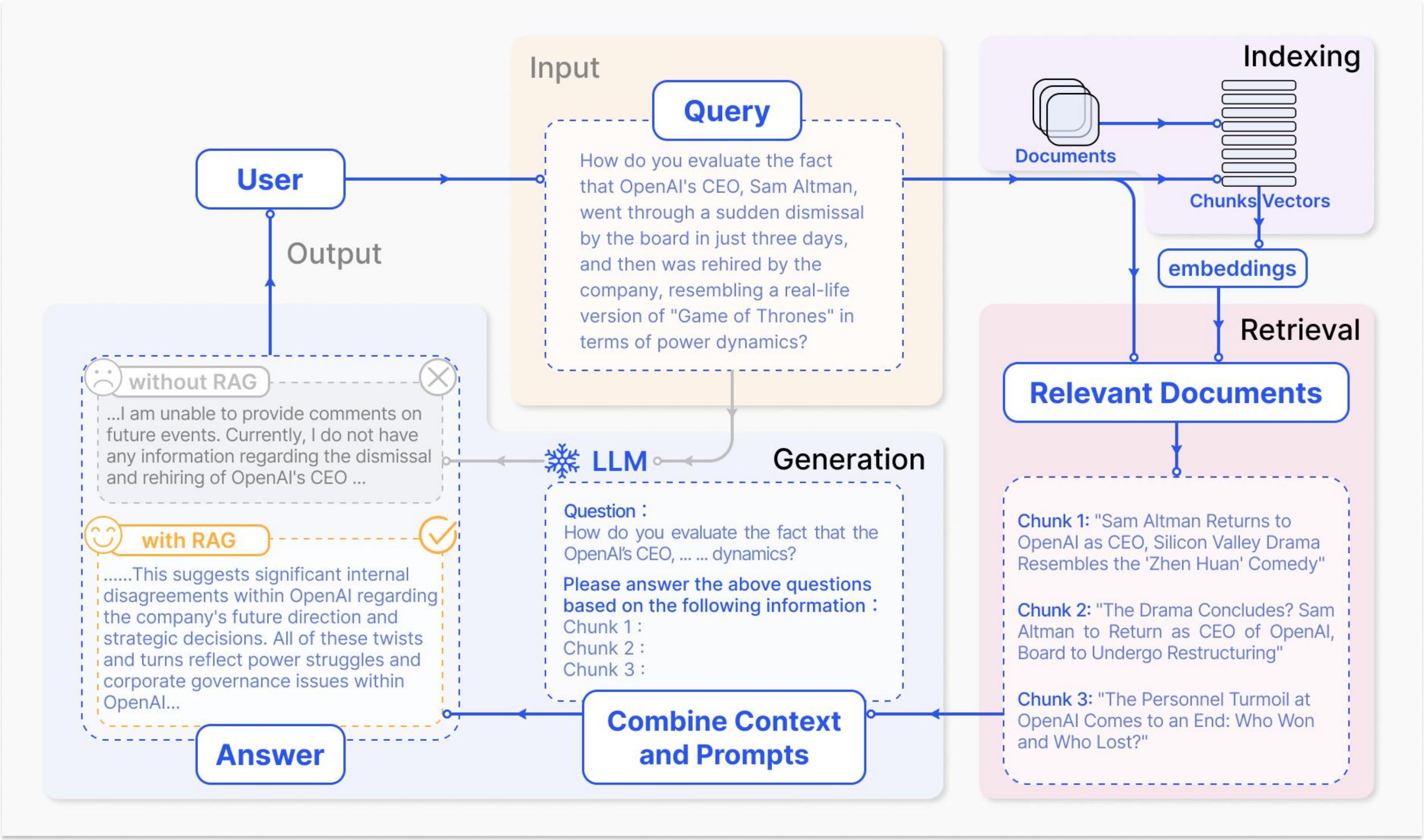


Retrieval-Augmented Generation (RAG)

When answering questions or generating text, it first retrieves relevant information from a large number of documents, and then LLMs generates answers based on this information.

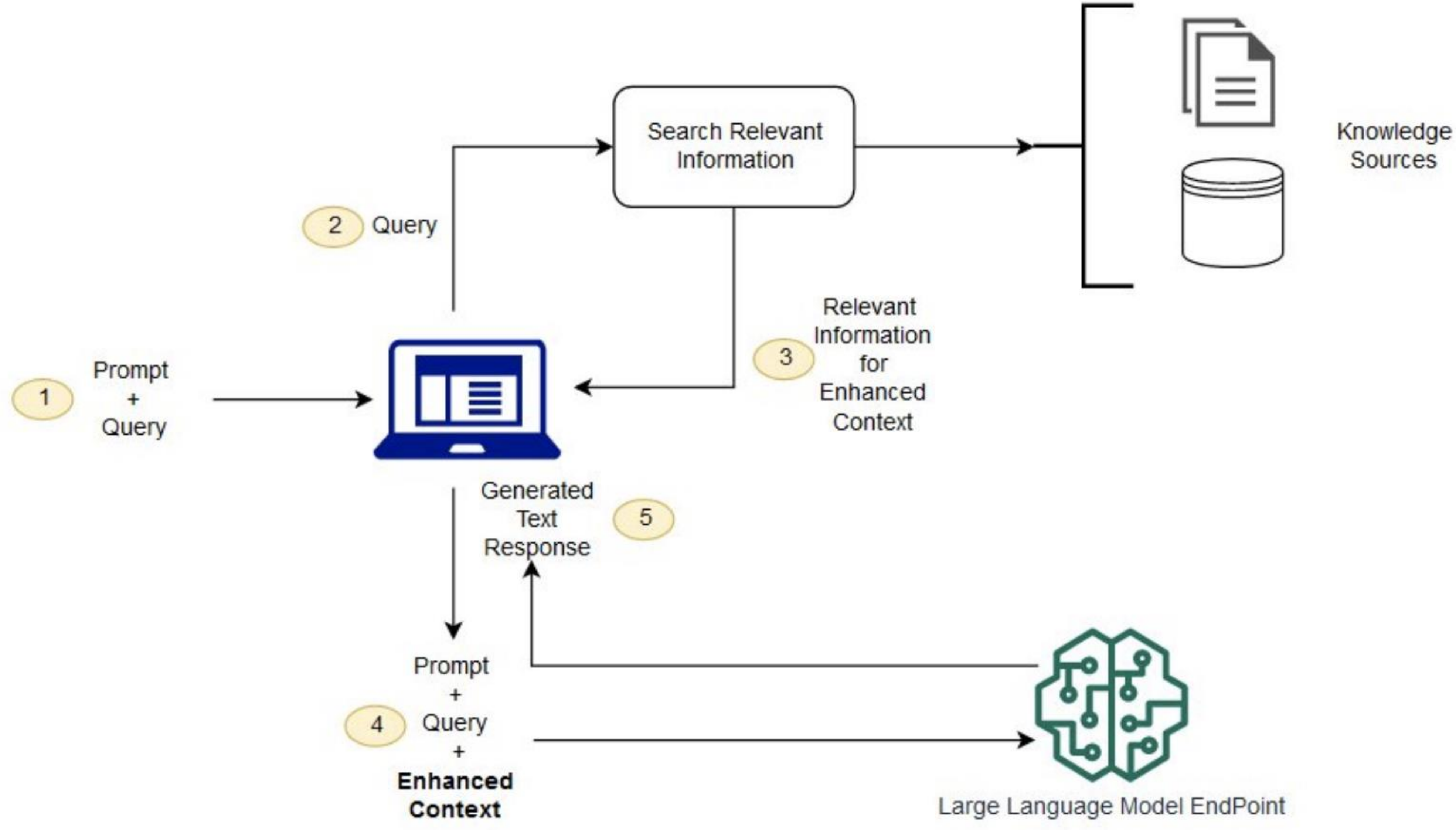
By attaching a external knowledge base, there is no need to retrain the entire large model for each specific task.

The RAG model is especially suitable for knowledge-intensive tasks.



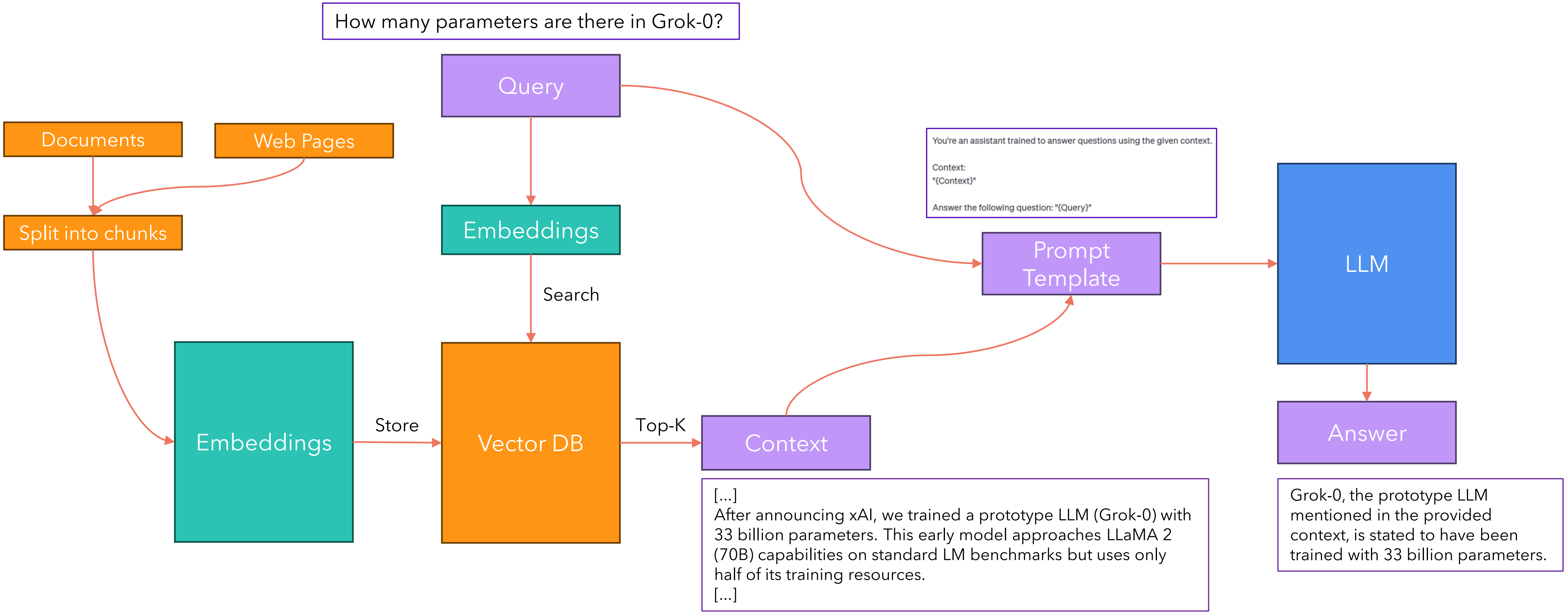
A typical case of RAG

The following diagram shows the conceptual flow of using RAG with LLMs.



https://aws.amazon.com/what-is/retrieval-augmented-generation/

QA with Retrieval Augmented Generation



Example

You

I think Vin Diesel has been a voice actor for several characters in TV series, do you know what their names are? Please list all of them.

ChatGPT

Vin Diesel has lent his voice to several characters across various TV series and shows. Here's a list of the characters he's voiced:

1. **Groot** in "Guardians of the Galaxy Vol. 3" (2023 Movie), "The Guardians of the Galaxy Holiday Special" (2022 TV Show), "I Am Groot" (2022 Short), "Thor: Love and Thunder" (2022 Movie), "Avengers: Endgame" (2019 Movie), "Ralph Breaks the Internet: Wreck-It Ralph 2" (2018 Movie), "Avengers: Infinity War" (2018 Movie), "Guardians of the Galaxy Vol. 2" (2017 Movie), and "Guardians of the Galaxy" (2014 Movie).
2. **Santiago** in "ARK: The Animated Series" (2023 TV Show).
3. **Dominic Toretto** in "Fast & Furious: Spy Racers" (2019 TV Show).

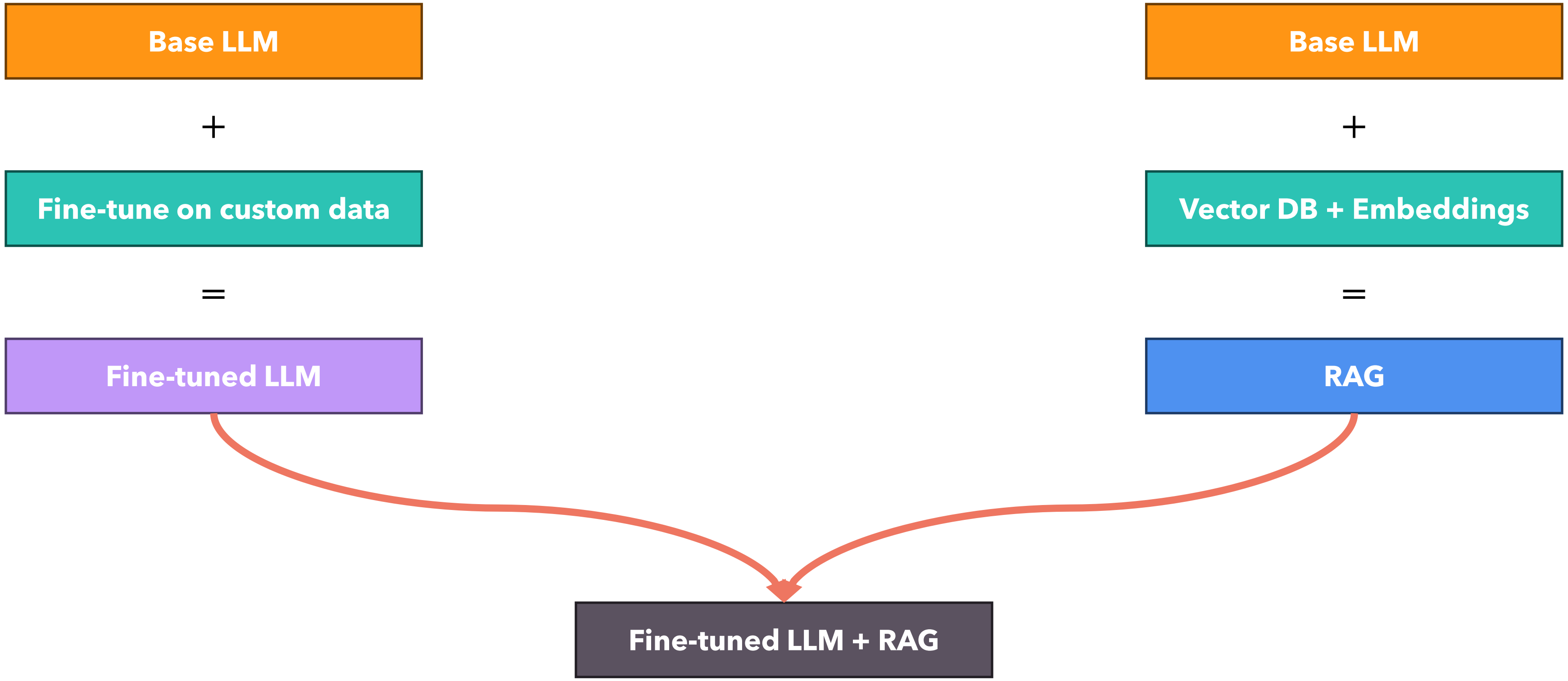
https://www.behindthevoiceactors.com/Vin-Diesel/

Additionally, though the website for "Big Mouth" lists Vin Diesel, it appears to be a misunderstanding or an error, as Nick Kroll is credited as the voice of Vin Diesel in that show.

https://www.behindthevoiceactors.com/tv-shows/Big-Mouth/Vin-Diesel/

These roles showcase Diesel's versatility and contribution to voice acting, particularly in animated series and specials, where he brings iconic characters to life with his distinctive voice.

Strategies to teach new concepts to LLM



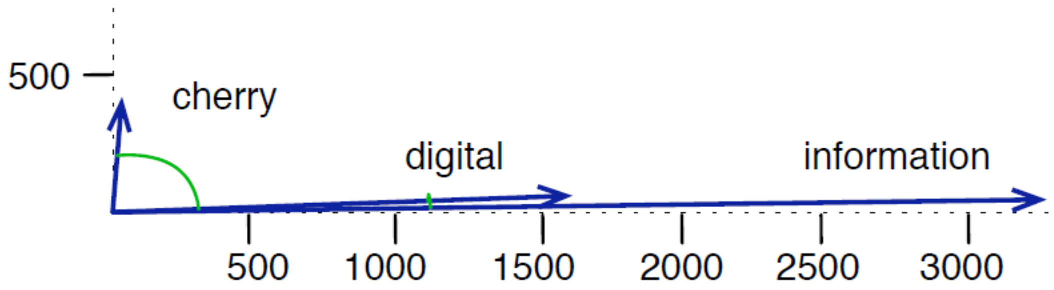
RAG vs Fine-tuning

Feature Comparison	RAG	Fine-Tuning
Knowledge Updates	Directly updating the retrieval knowledge base ensures that the information remains current without the need for frequent retraining, making it well-suited for dynamic data environments.	Stores static data, requiring retraining for knowledge and data updates.
External Knowledge	Proficient in leveraging external resources, particularly suitable for accessing documents or other structured/unstructured databases.	Can be utilized to align the externally acquired knowledge from pretraining with large language models, but may be less practical for frequently changing data sources.
Data Processing	Involves minimal data processing and handling.	Depends on the creation of high-quality datasets, and limited datasets may not result in significant performance improvements.
Model Customization	Focuses on information retrieval and integrating external knowledge but may not fully customize model behavior or writing style.	Allows adjustments of LLM behavior, writing style, or specific domain knowledge based on specific tones or terms.
Interpretability	Responses can be traced back to specific data sources, providing higher interpretability and traceability.	Similar to a black box, it is not always clear why the model reacts a certain way, resulting in relatively lower interpretability.
Computational Resources	Depends on computational resources to support retrieval strategies and technologies related to databases. Additionally, it requires the maintenance of external data source integration and updates.	The preparation and curation of high-quality training datasets, defining fine-tuning objectives, and providing corresponding computational resources are necessary.
Latency Requirements	Involves data retrieval, which may lead to higher latency.	LLM after fine-tuning can respond without retrieval, resulting in lower latency.
Reducing Hallucinations	Inherently less prone to hallucinations as each answer is grounded in retrieved evidence.	Can help reduce hallucinations by training the model based on specific domain data but may still exhibit hallucinations when faced with unfamiliar input.
Ethical and Privacy Issues	Ethical and privacy concerns arise from the storage and retrieval of text from external databases.	Ethical and privacy concerns may arise due to sensitive content in the training data.

Retrieval Methods

Why do we use vectors to represent words?

Given the words “**cherry**”, “**digital**” and “**information**”, if we represent the embedding vectors using only 2 dimensions (X, Y) and we plot them, we hope to see something like this: the angle between words with similar meaning is small, while the angle between words with different meaning is big. So, the embeddings “capture” the meaning of the words they represent by projecting them into a high-dimensional space.



Source: Speech and Language Processing 3rd Edition Draft, Dan Jurafsky and James H. Martin

We commonly use the **cosine similarity**, which is based on the **dot product** between the two vectors.

Sparse Retrieval

- Express the query and document as a sparse word frequency vector (usually normalized by length)

q=what is nlp

d_1 = what is life ? candy is life !

d_2 = nlp is an acronym for natural language processing

d_3 = I like to do good research on nlp

$$
\begin{array}{c}
\text{what} \\ \text{candy} \\ \text{nlp} \\ \text{is} \\ \text{language} \\ \dots
\end{array}
\begin{pmatrix} 0.33 \\ 0 \\ 0.33 \\ 0.33 \\ 0 \\ \dots \end{pmatrix}
\quad
\begin{pmatrix} 0.25 \\ 0.125 \\ 0 \\ 0.25 \\ 0 \\ \dots \end{pmatrix}
\quad
\begin{pmatrix} 0 \\ 0 \\ 0.125 \\ 0.125 \\ 0 \\ \dots \end{pmatrix}
\quad
\begin{pmatrix} 0 \\ 0 \\ 0.125 \\ 0 \\ 0 \\ \dots \end{pmatrix}
$$

$q^*d_1 = 0.165$ $\quad$ $q^*d_2 = 0.0825$ $\quad$ $q^*d_3 = 0.0413$

- Find the document with the highest inner-product or cosine similarity in the document collection

Term Weighting

(See Manning et al. 2009)

- Some terms are more important than others; low-frequency words are often more important
- Term frequency - in-document frequency (TF-IDF)

$$\mathrm{TF}(t,d) = \frac{\mathrm{freq}(t,d)}{\sum_{t'} \mathrm{freq}(t',d)} \qquad \mathrm{IDF}(t) = \log\left(\frac{|D|}{\sum_{d' \in D} \delta(\mathrm{freq}(t,d') > 0)}\right)$$

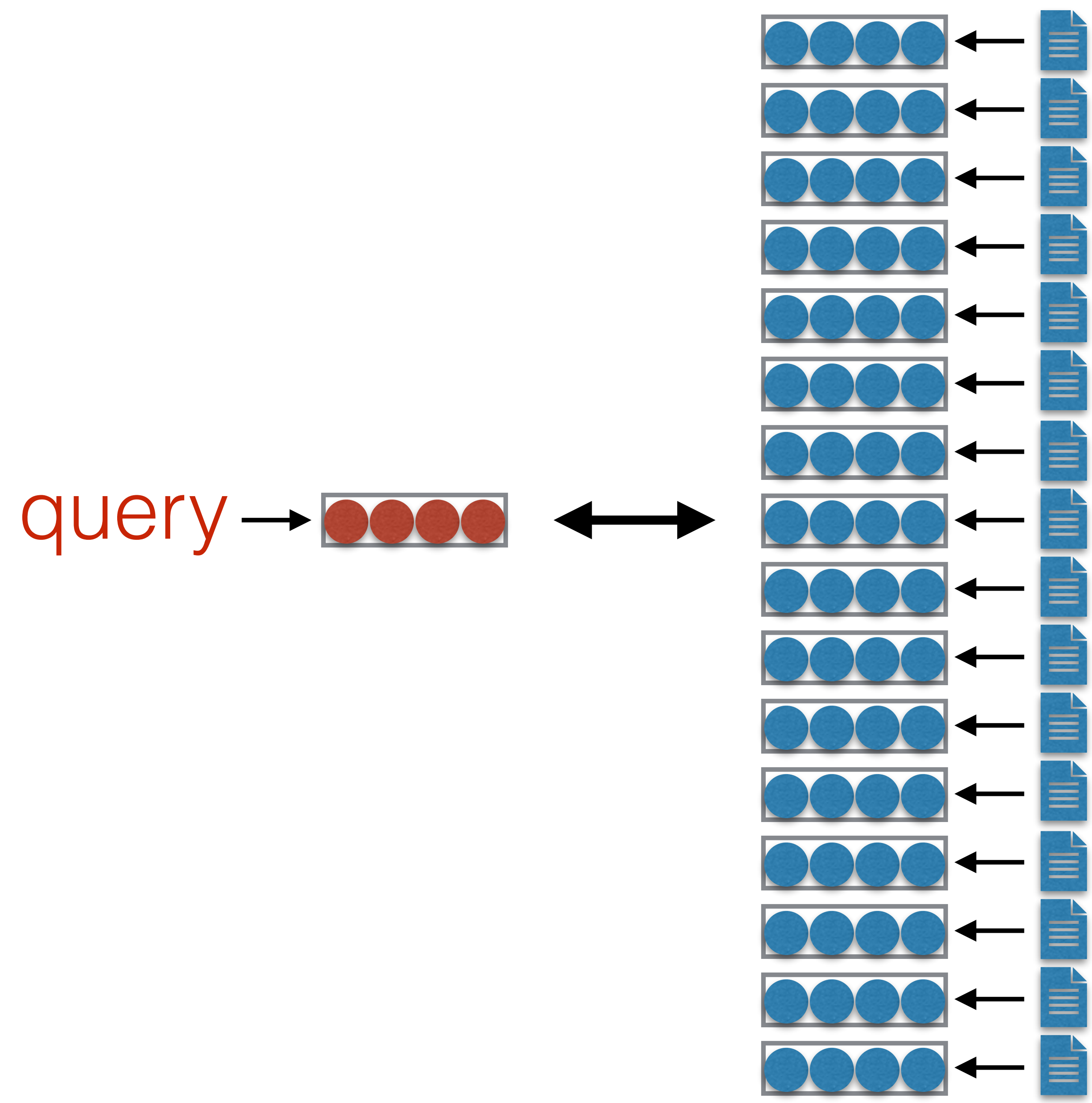
$$\text{TF-IDF}(t,d) = \mathrm{TF}(t,d) \times \mathrm{IDF}(t)$$

- BM25: TF term similar to smoothed count-based LMS

$$\text{BM-25}(t,d) = \mathrm{IDF}(t) \cdot \frac{\mathrm{freq}(t,d) \cdot (k_1 + 1)}{\mathrm{freq}(t,d) + k_1 \cdot \left(1 - b + b \cdot \frac{|d|}{\mathrm{avgdl}}\right)}$$

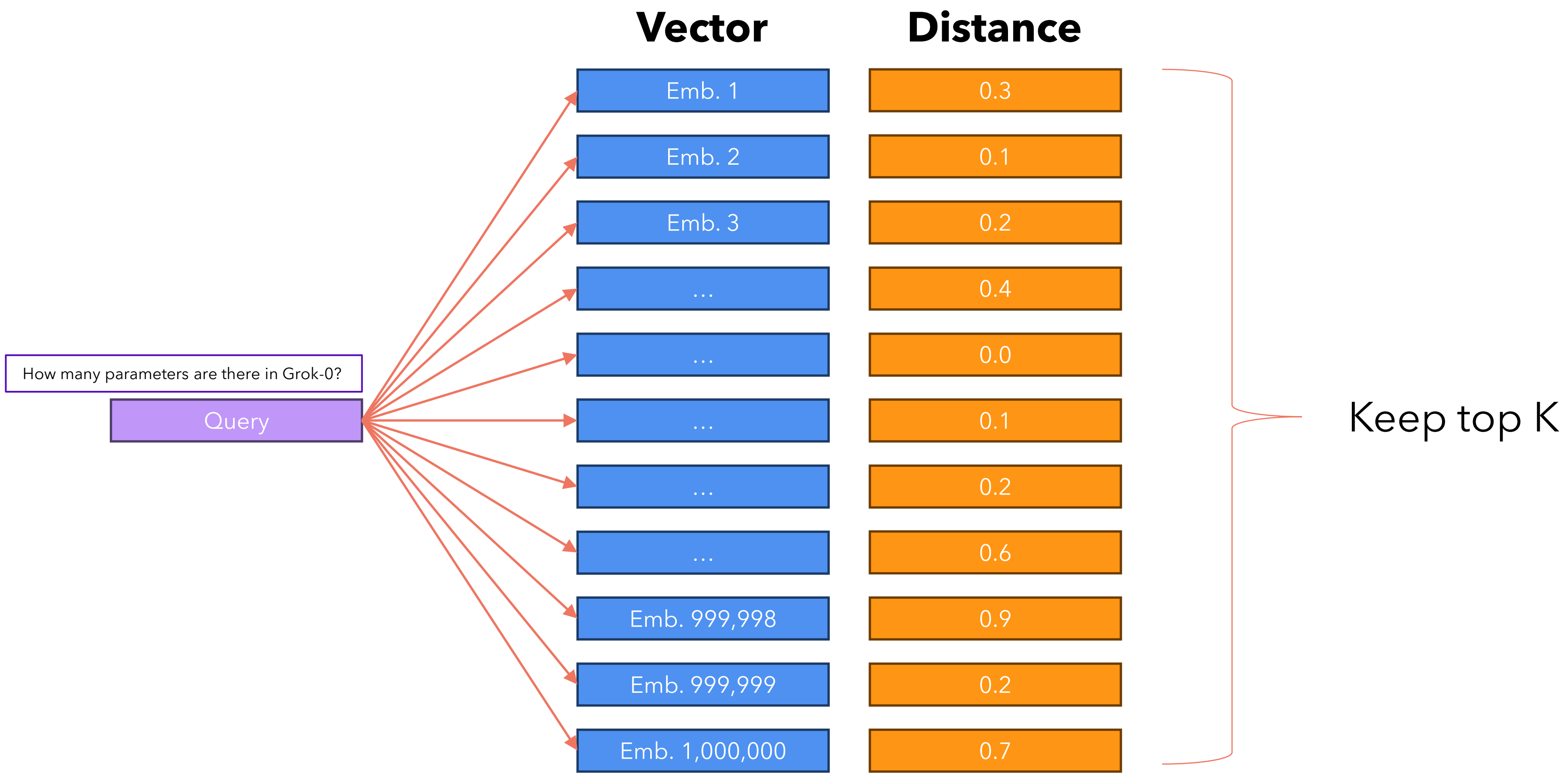
Dense Retrieval

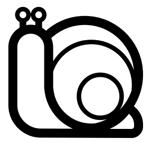
- Encode document/query and find nearest neighbor
- Can use:
    - Out-of-the-box embeddings
    - Learned embeddings



K-NN: a naïve approach

Imagine we want to search for the query in our database: a simple way would be comparing the query with all the vectors, sorting them by distance, and keeping the top K.



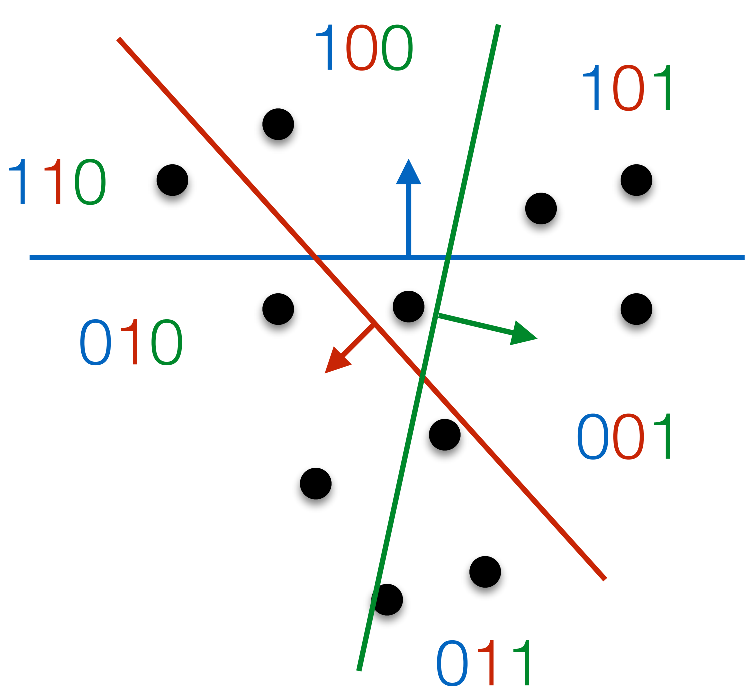


If there are N embedding vectors and each has D dimensions, the computational complexity is in the order of **O(N*D), too slow!**

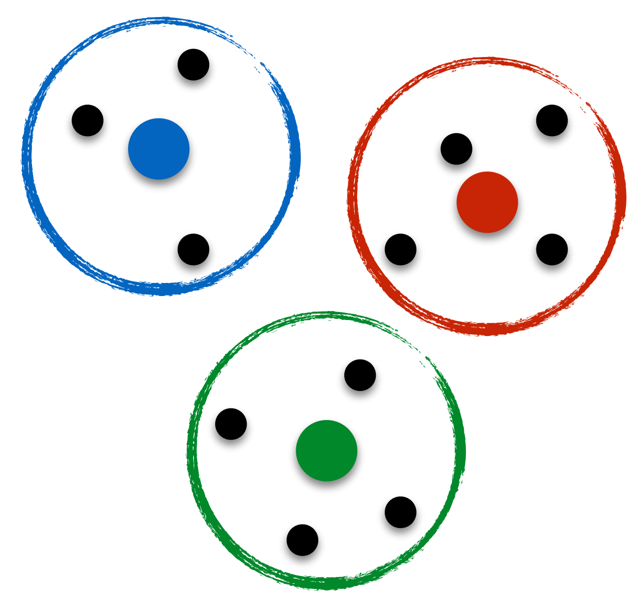
Approximate Nearest Neighbor Search

- Methods to retrieve embeddings in sub-linear time

Locality sensitive hashing: make partitions in continuous space, use like inverted index



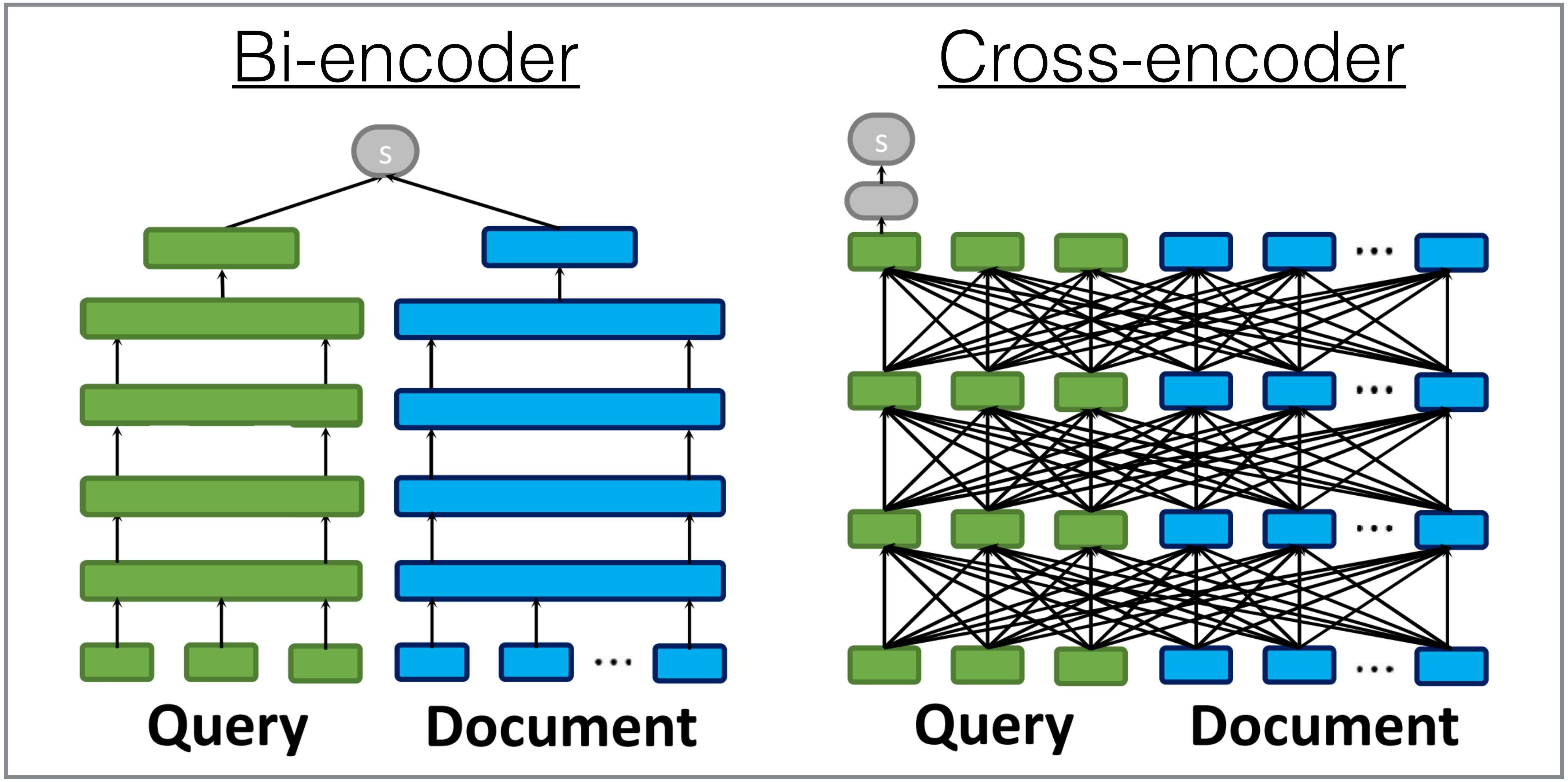
Graph-based search: create "hubs" and search from there



- Software: FAISS, ChromaDB

Cross-encoder Reranking

- Jointly encode both queries and documents using neural model (Nogueira et al. 2019)

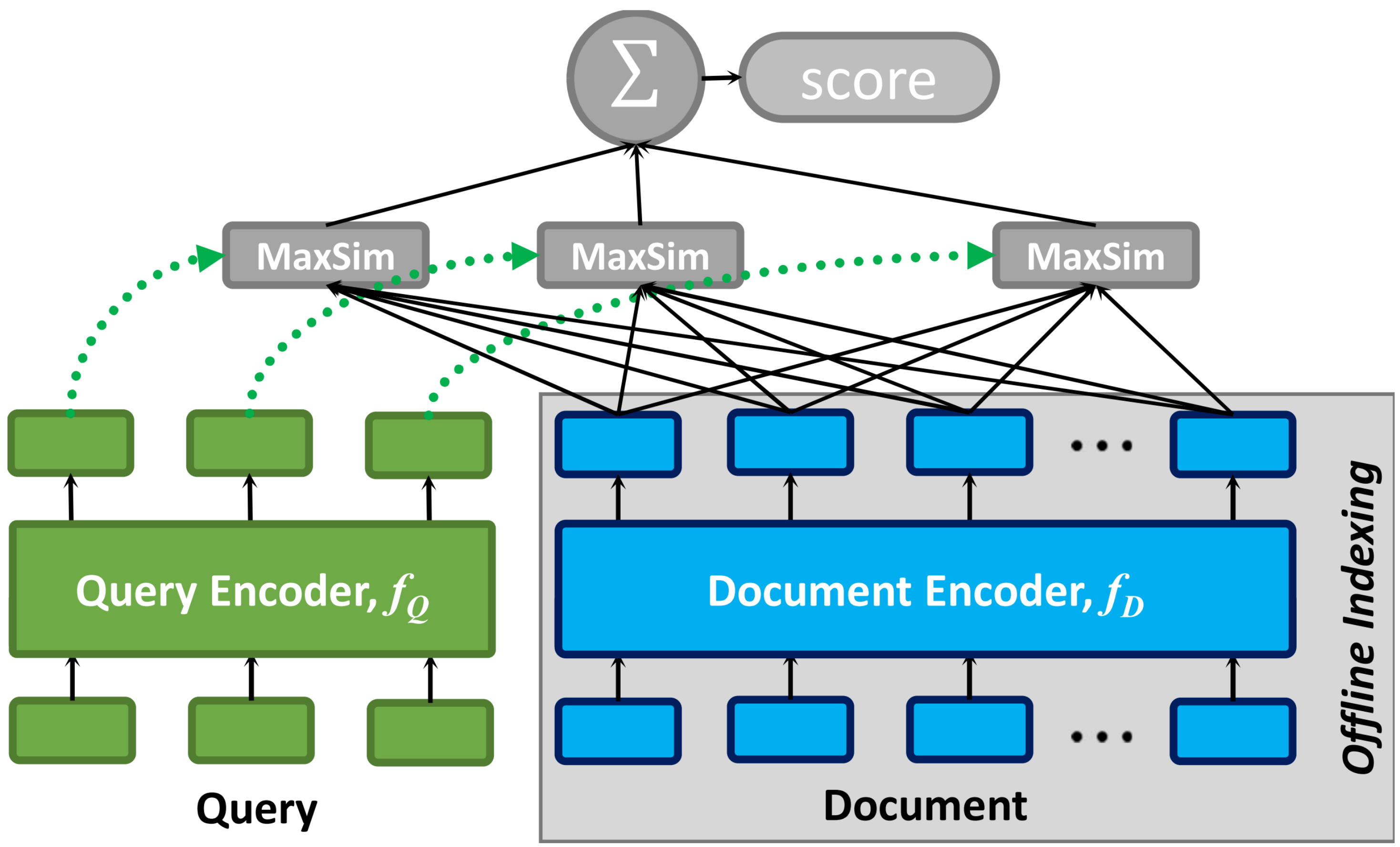


- Precludes approximate nearest neighbor lookup, so can only be used on small number of candidates

Figure from Khattab et al. (2020)

Token-level Dense Retrieval

- ColBERT (Khattab et al. 2020) use contextual representations of all query and document tokens to compute retrieval score.



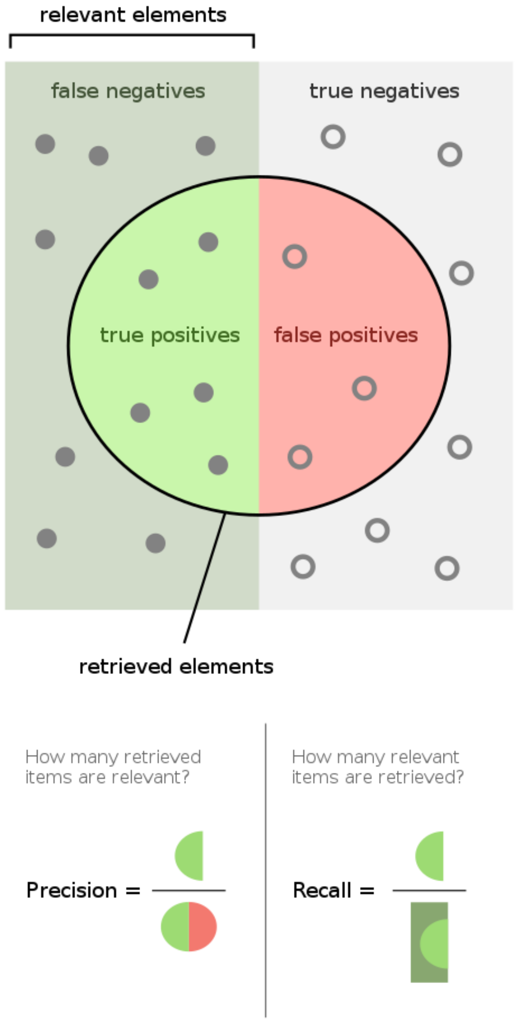
- Significantly more effective (but more costly) than single-vector retrieval

Similarity Search: let's trade precision for speed

The naïve approach we used before, always produces accurate results, since it compares the query with all the stored vectors, but what if we reduced the number of comparison, but still obtain accurate results with high probability?

The metric we usually care about in Similarity Search is recall.

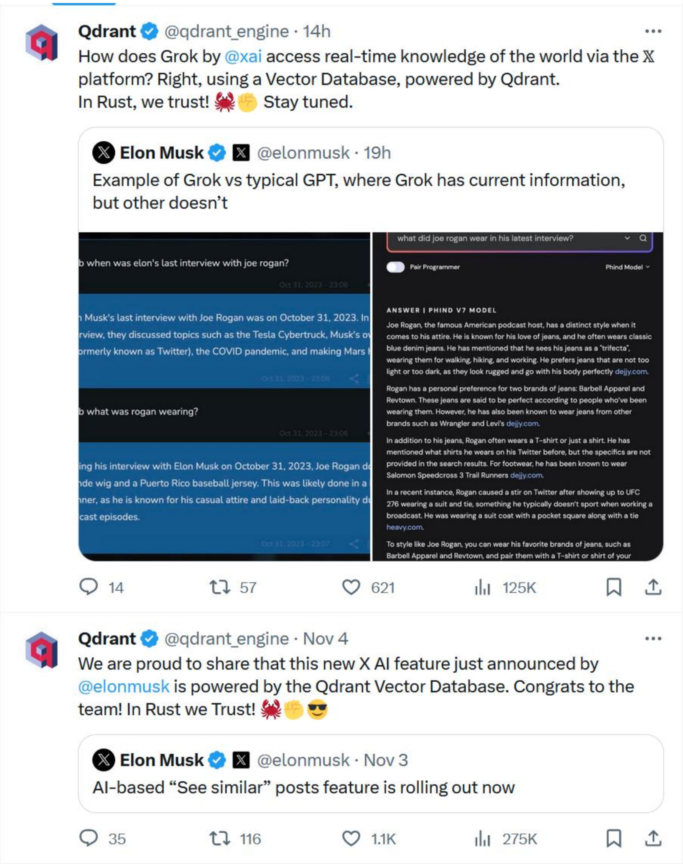
we will explore an algorithm for Approximate Nearest Neighbors, called **Hierarchical Navigable Small Worlds (HNSW)**.

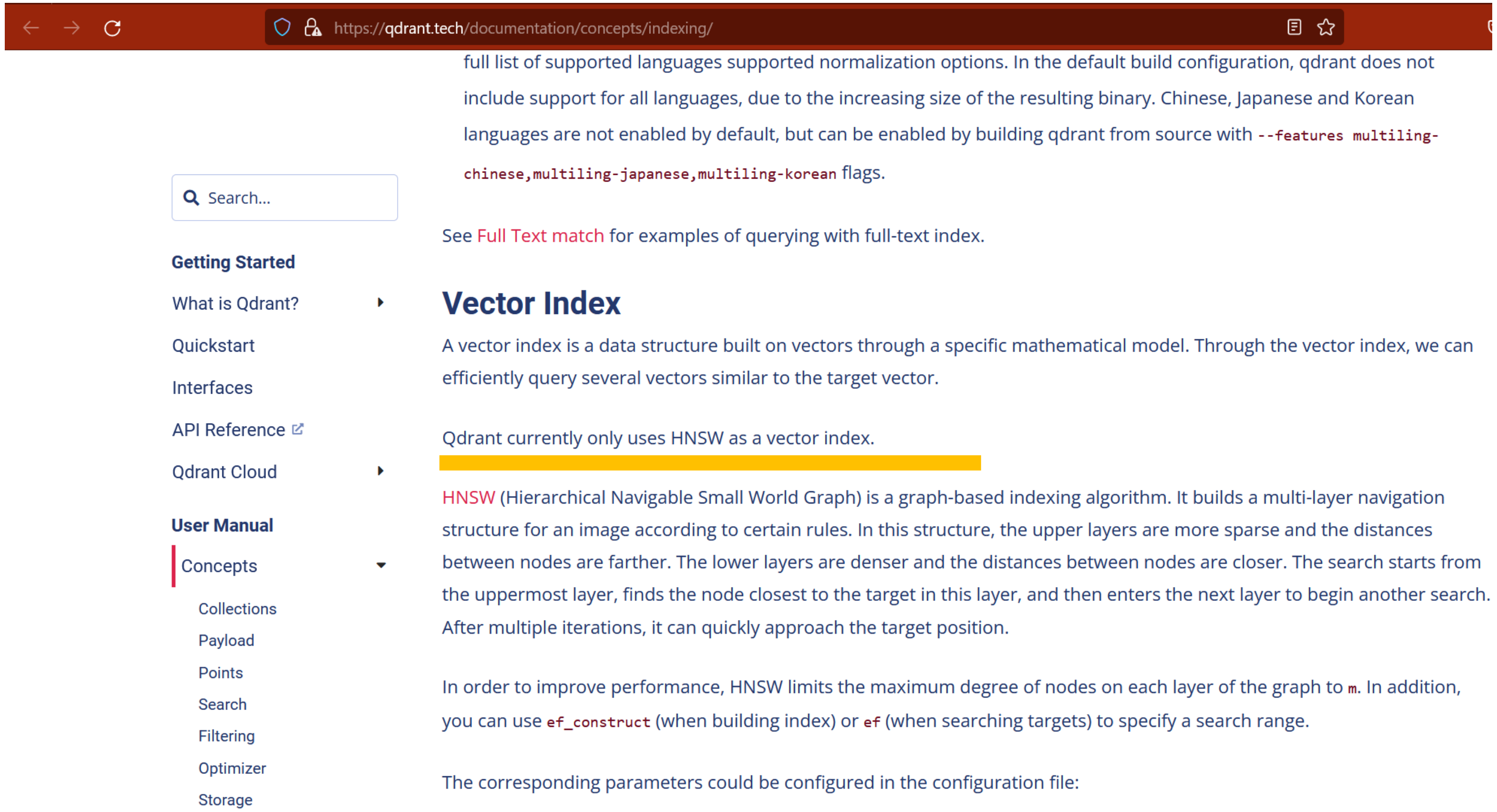


Source: Wikipedia

HNSW in the real world

It is the same algorithm that powers **Qdrant**, the open source Vector DB used by **Twitter's** (X) **Grok** LLM, which can access tweets in real time.





HNSW: idea #1

HNSW is an evolution of the **Navigable Small Worlds** algorithm for Approximate Nearest Neighbors, which is based on the concept of **Six Degrees of Separation**.

Milgram's experiment aimed to test the social connections among people in the United States. The participants, who were initially located in Nebraska and Kansas, were given a letter to be delivered to a specific person in Boston. However, they were not allowed to send the letter directly to the recipient. Instead, they were instructed to send it to someone they knew on a first-name basis, who they believed might have a better chance of knowing the target person.

At the end of Milgram's small-world experiment, Milgram found that most of the letters reached the final recipient in five or six steps, creating the concept that people all over the world are all connected by six degrees of separation.

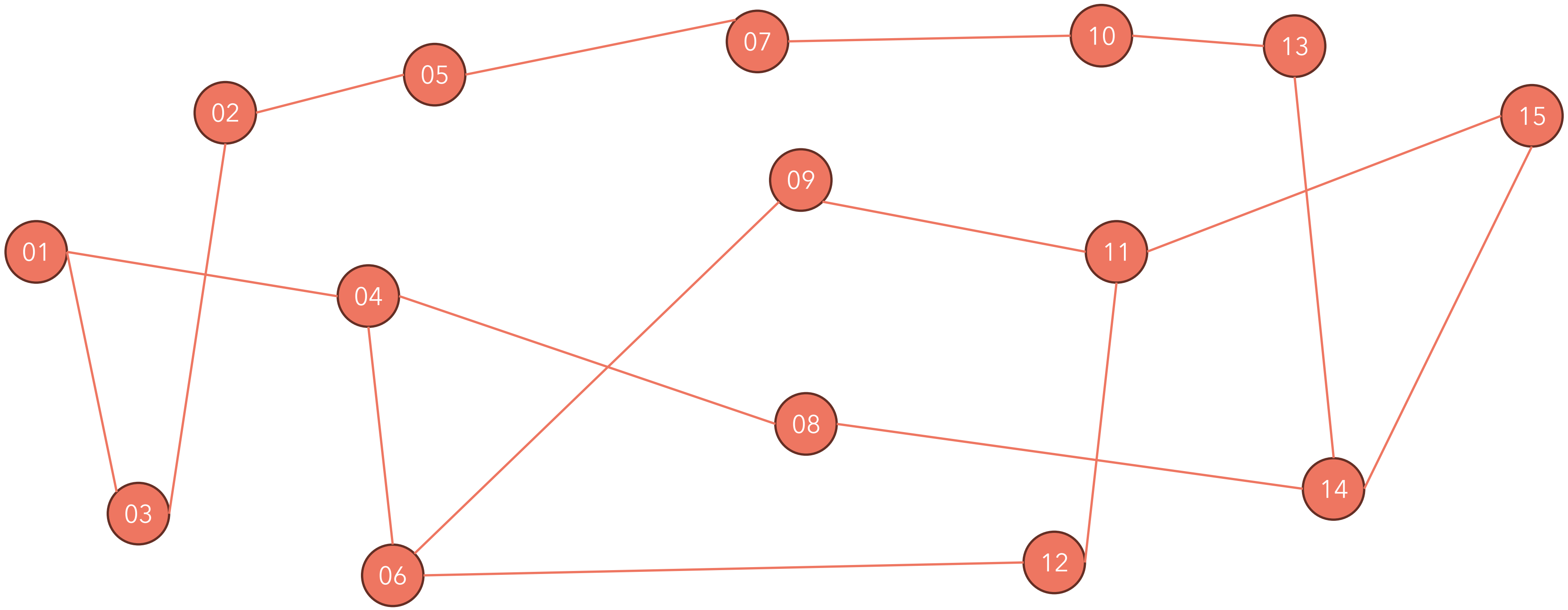
Facebook found in 2016 that its 1.59 billion active users were connected on average by 3.5 degrees of separation: https://research.facebook.com/blog/2016/02/three-and-a-half-degrees-of-separation/

This means that you and Mark Zuckerberg are only 3.5 connections apart!

Navigable Small Worlds

The NSW algorithm builds a graph that – just like Facebook friends – connects close vectors with each other but keeping the total number of connections small. For example, every vector may be connected to up to 6 other vectors (to mimic the Six Degrees of Separation).

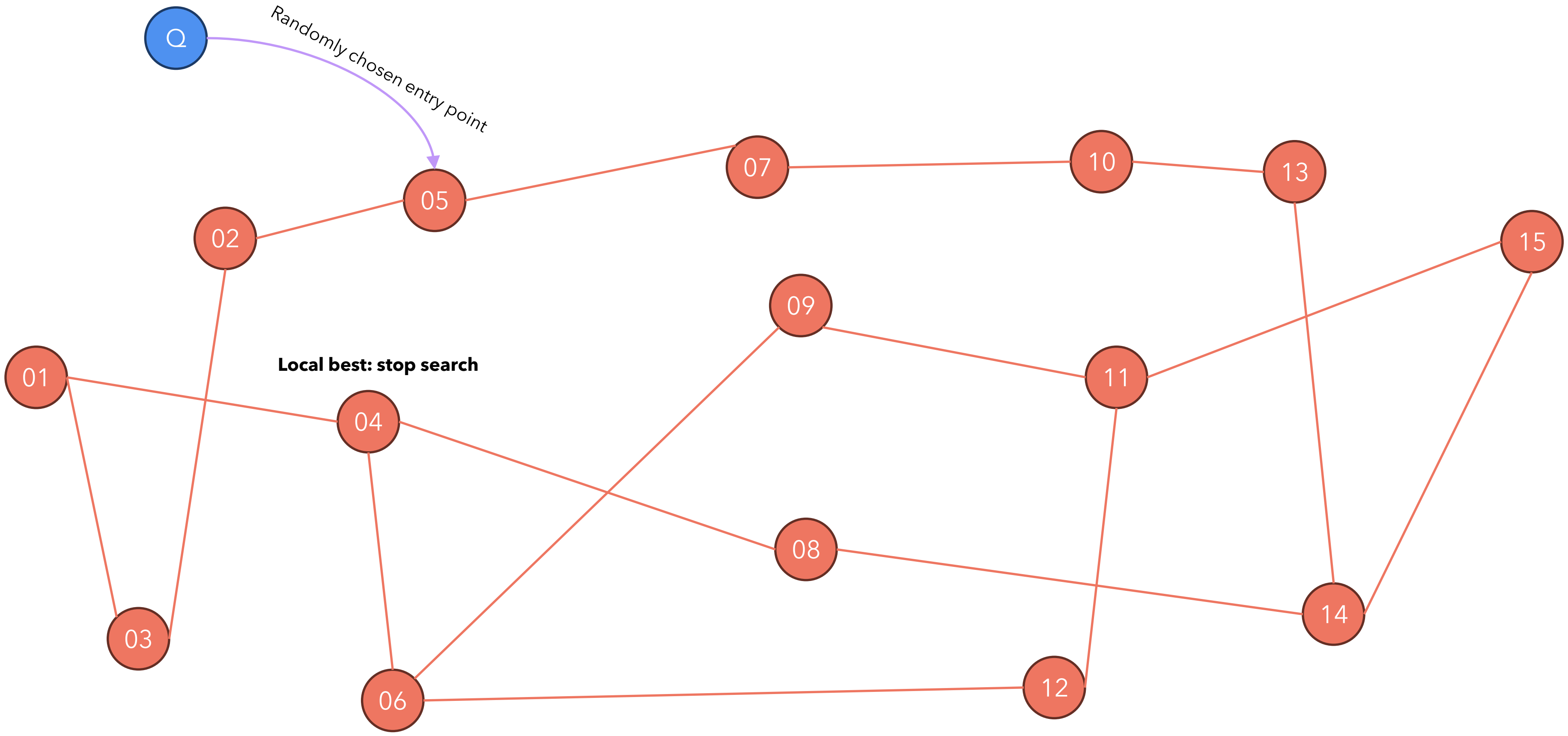
Node	Text
01	[…] The Transformer is a model […]
02	[…] Diagnose cancer with AI […]
03	[…] A transformer-based model […]
04	[…] The Transformer has 6 layers […]
05	[…] An MRI machine that costs 1$ […]
06	[…] The dot-product is a […]
07	[…] Big-Pharma is not so big […]
08	[…] Cross-Attention is a great […]
09	[…] To solve an ODE […]
10	[…] We are aging too fast […]
11	[…] Open-source models like […]
12	[…] MathBERT: a new model […]
13	[…] AI to control aging […]
14	[…] Attention is all you need […]
15	[…] LLaMA 2 has 7B params […]



Navigable Small Worlds: searching for K-NN

Given the following query: *"How many Encoder layers are there in the Transformer model?"*
How does the algorithm find the K Nearest Neighbors?

Node	Text
01	[…] The Transformer is a model […]
02	[…] Diagnose cancer with AI […]
03	[…] A transformer-based model […]
04	[…] The Transformer has 6 layers […]
05	[…] An MRI machine that costs 1$ […]
06	[…] The dot-product is a […]
07	[…] Big-Pharma is not so big […]
08	[…] Cross-Attention is a great […]
09	[…] To solve an ODE […]
10	[…] We are aging too fast […]
11	[…] Open-source models like […]
12	[…] MathBERT: a new model […]
13	[…] AI to control aging […]
14	[…] Attention is all you need […]
15	[…] LLaMA 2 has 7B params […]



We repeat the search with randomly chosen starting points and then keep the top K among all the visited nodes.

Navigable Small Worlds: inserting a new vector

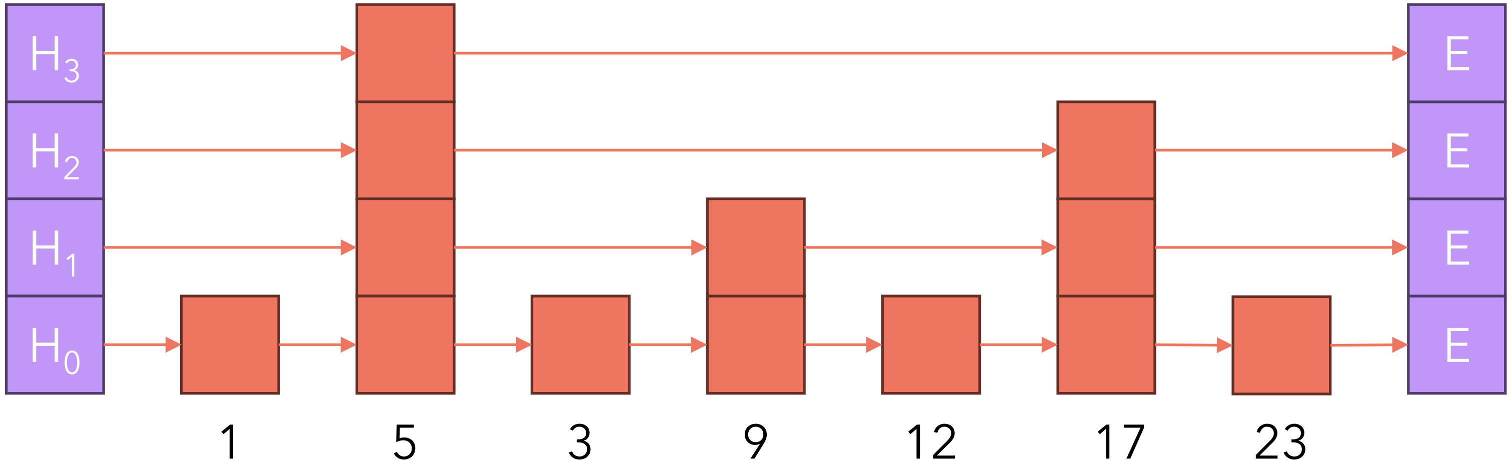
We can insert a new vector by searching the top KNN with the searching algorithm described before and making an edge between the vector and the top K results.

HNSW: idea #2

To go from NSW (Navigable Small Worlds) to HNSW (Hierarchical Navigable Small Worlds), we need to introduce the algorithm behind the data structure known as Skip-List.

The skip list is a data structure that maintains a sorted list and allows search and insertion with an average of $O(\log N)$ time complexity.

Let's search the number **9**

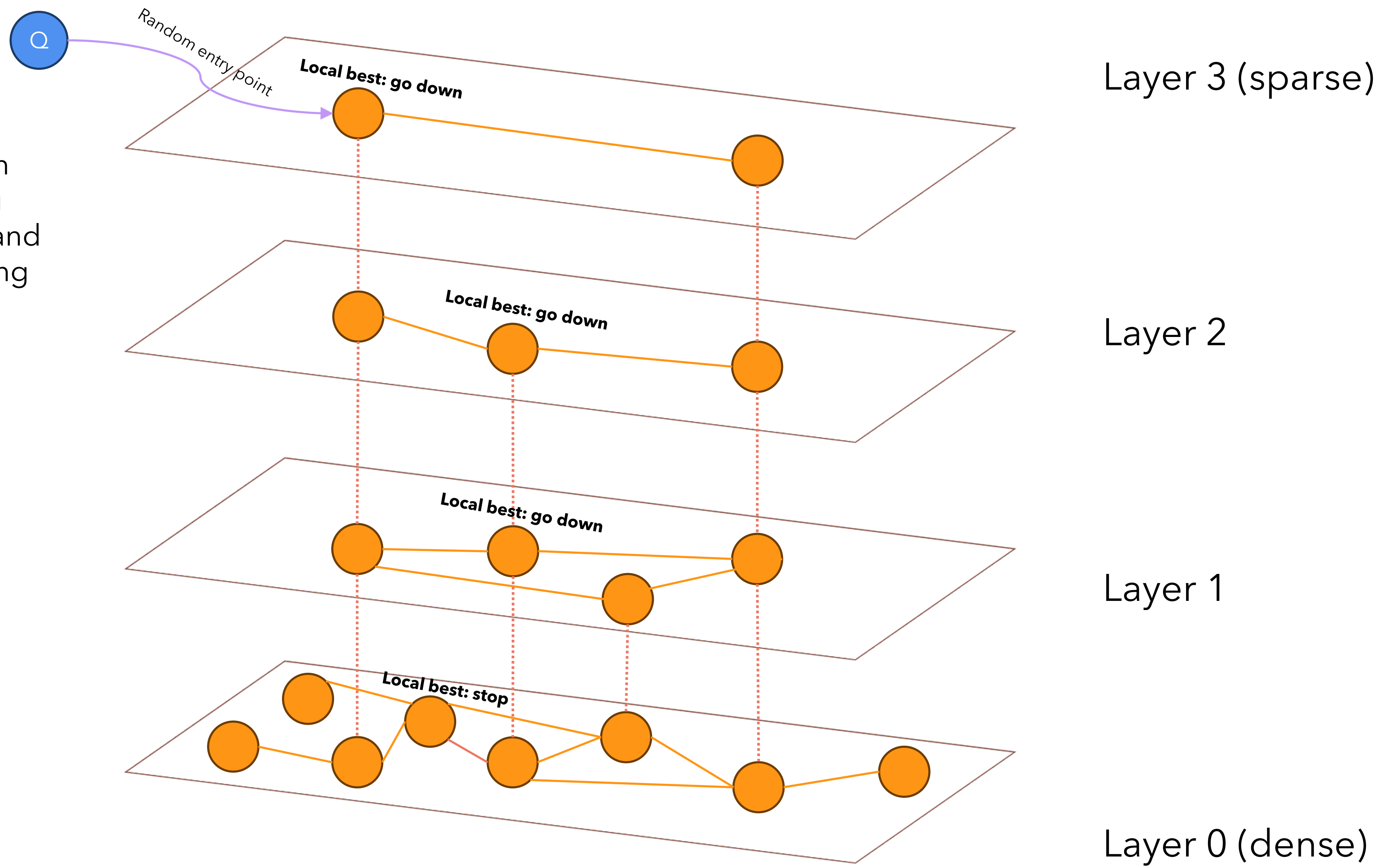


HNSW: Hierarchical Navigable Small Worlds

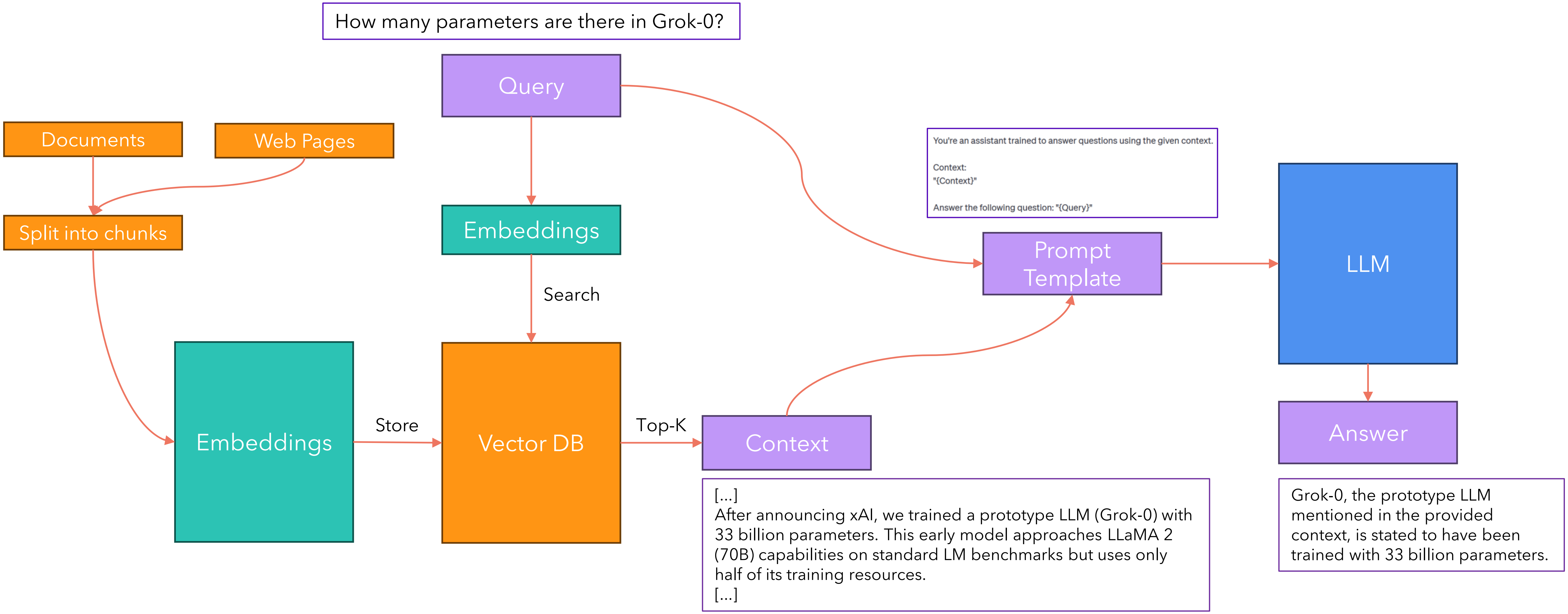
HNSW: Hierarchical Navigable Small Worlds

Let's search!

We repeat the search with randomly chosen starting points (on the top layer) and then keep the top K among all the visited nodes.



QA with Retrieval Augmented Generation



When do we Retrieve?

- **Once, at the beginning of generation**
  - Default method used by most systems (Lewis et al. 2020)
- **Several times during generation, as necessary**
  - Generate a search token (Schick et al. 2023)
  - Search when the model is uncertain (Jiang et al. 2023)
- **Every token**
  - Find similar final embeddings (Khandelwal et al. 2019)
  - Approximate attention with nearest neighbors (Bertsch et al. 2023)

Triggering Retrieval w/ Tokens

- Toolformer (Schick et al. 2023) generates tokens that trigger retrieval (or other tools)
- Training is done in an iterative manner - generate and identify successful retrievals

The New England Journal of Medicine is a registered trademark of **[QA("Who is the publisher of The New England Journal of Medicine?") → Massachusetts Medical Society]** the MMS.

Out of 1400 participants, 400 (or **[Calculator(400 / 1400) → 0.29]** 29%) passed the test.

The name derives from "la tortuga", the Spanish word for **[MT("tortuga") → turtle]** turtle.

The Brown Act is California's law **[WikiSearch("Brown Act") → The Ralph M. Brown Act is an act of the California State Legislature that guarantees the public's right to attend and participate in meetings of local legislative bodies.]** that requires legislative bodies, like city councils, to hold their meetings open to the public.

Triggering Retrieval w/ Uncertainty

- FLARE (Jiang et al. 2023) tries to generate content, then does retrieval if LM certainty is low