How we made AI chatbots lie less

Project C Survival Squad:

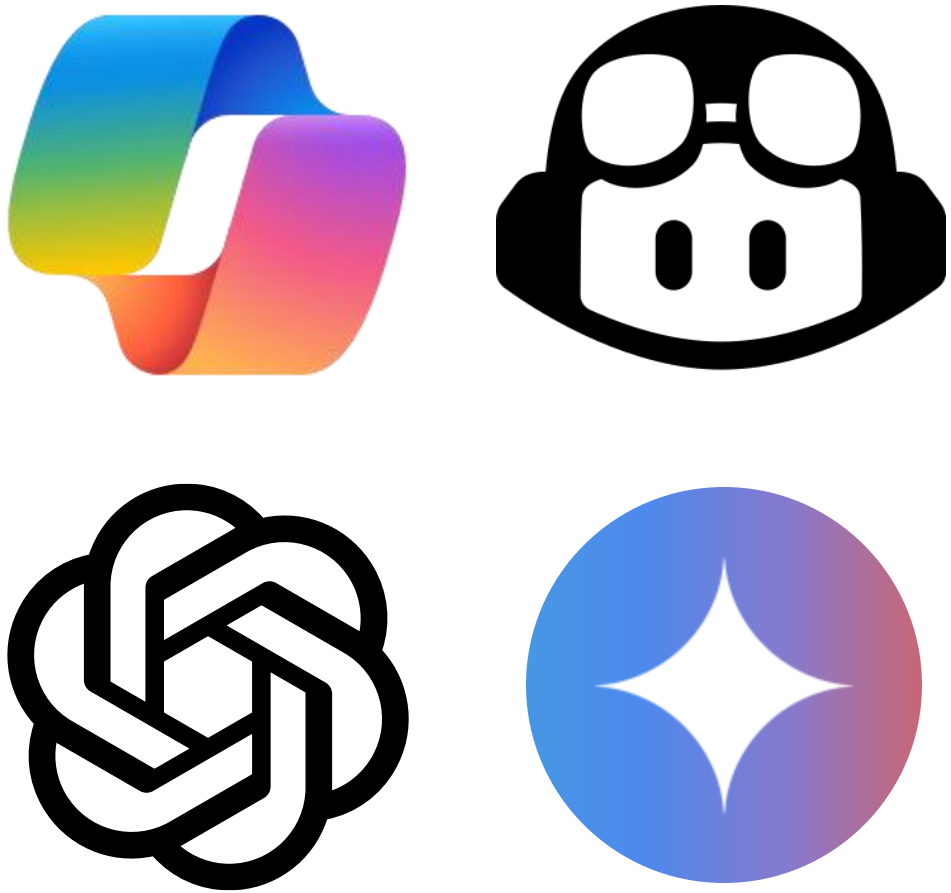
Alexander Small

Khanh Toan Nguyen

Matthew Crick

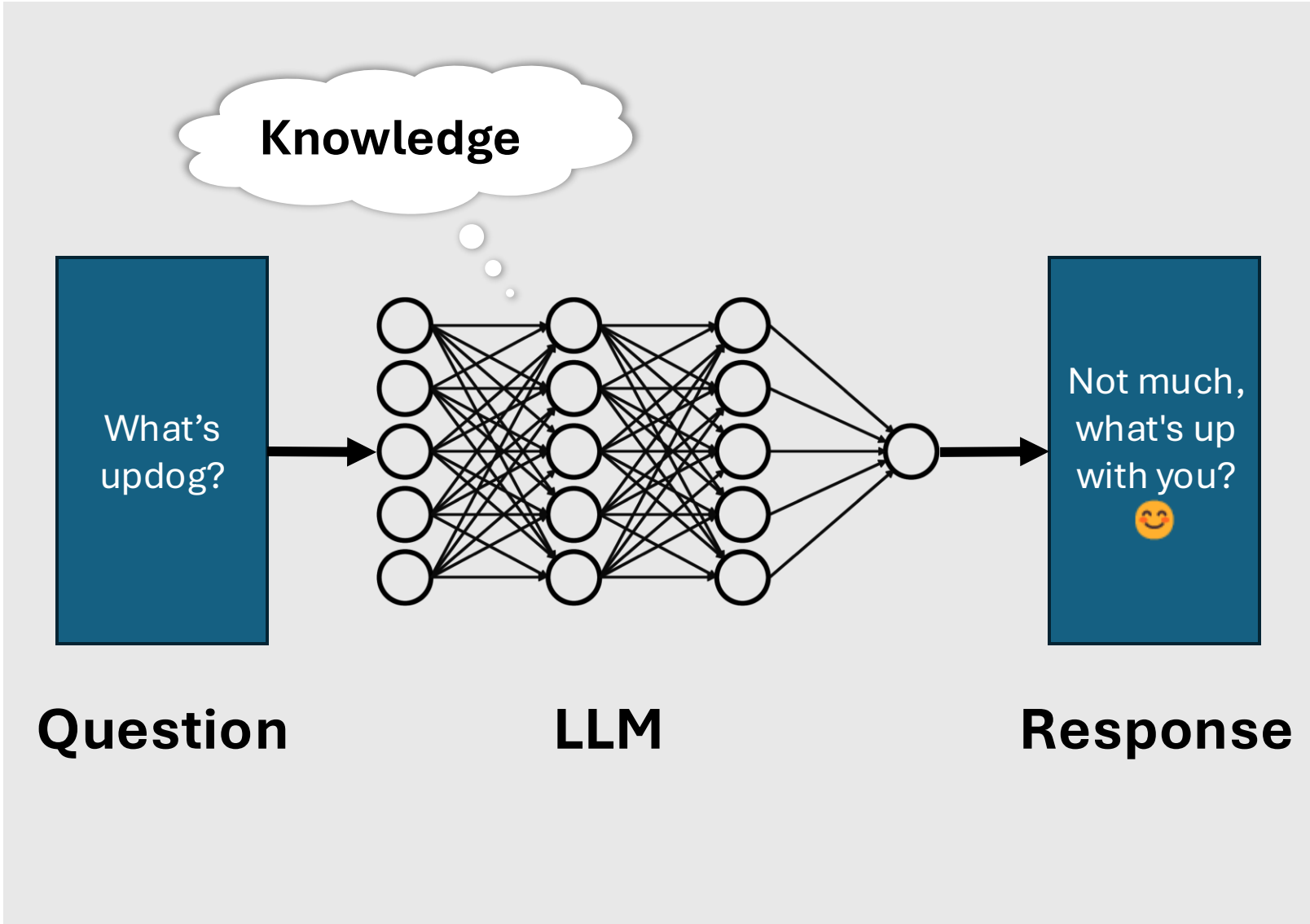
SWINBURNE UNIVERSITY OF TECHNOLOGY

Large Language Models are **everywhere**



How do Large Language Models *work*?

- LLMs take a user's question and **transform** it into a response.
- **LLMs** learn how written speech works using **deep learning**.
- LLMs are trained on massive amounts of **textual data**.



How are LLMs trained?

- LLMs are trained on **massive, unsupervised datasets**.
  - Data scrapes of social media, e.g., Reddit, Stack Exchange.
  - Data scrapes of the internet.
  - Massive libraries of books.
- LLMs' knowledge is **static** based on the time it was trained.

Unsupervised Contrastive Pre-training on Massive Text Pairs mined from the Web

Common Crawl

WIKIPEDIA

reddit

stackoverflow

StackExchange

arXiv

Semantic Scholar

(Li et al. 2023)

What applications do LLMs have?

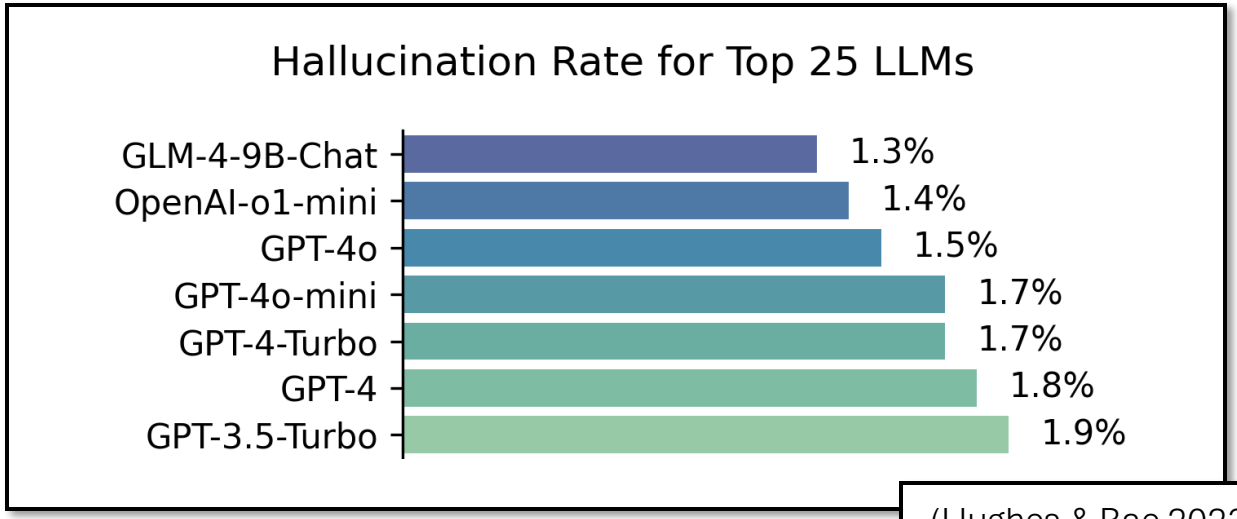
(Tonmoy et al. 2024)

- Providing mental health support (“**AI Therapy**”)
- Providing **legal advice**
- Summarising **medical records**
- Providing **customer support**
- Writing **financial analysis reports**
- Providing **personalised tutoring**

(Gates 2023)

Can we always trust what LLMs say?

- For many people, LLMs play a *key role in advising decisions and providing information*.
- But can we always trust what LLM's say? - do they always give *accurate* information?
- The evidence is unclear.



(Hughes & Bae 2023)

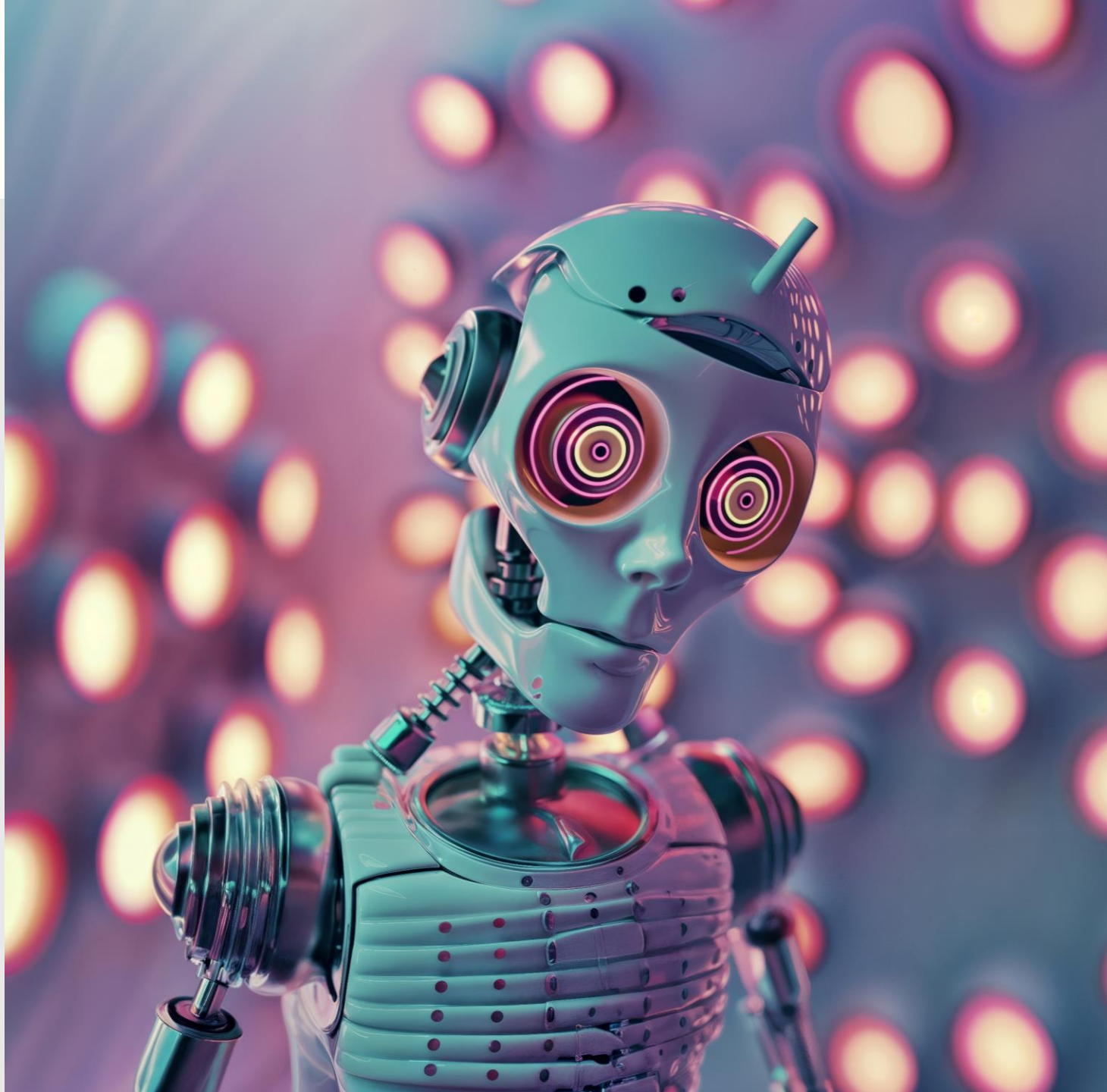
50 medical summaries by GPT-4o contained:

- 327 instances of medical event inconsistencies,
- 114 instances of incorrect reasoning and
- 3 instances of chronological inconsistencies.

(Adams 2024)

Are LLMs factual?

- LLMs are *only designed* to *predict the most likely next word* in a sentence.
- When responding, LLMs can **invent new information** that *sounds true* but *isn't, based on what it knows*.
- *This is called a* ***hallucination***.
- Hallucinations hold LLMs back from mainstream use.



The task

Alexa, make this LLM accurate!

Certainly Prof. Bao, calling all COS30018 students...

Alex

Toan

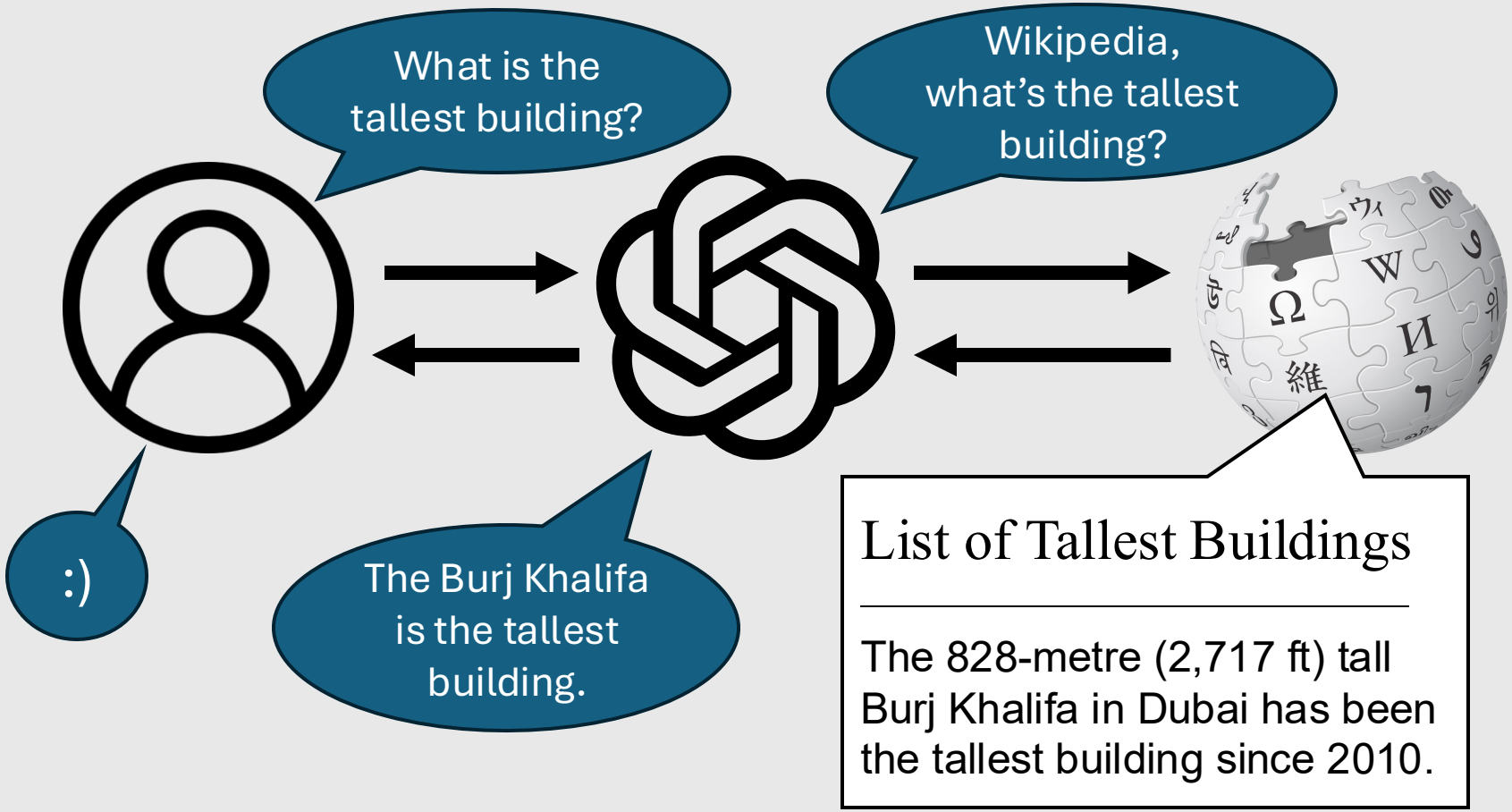
Matt

We can fix it!

How do we reduce LLM hallucinations ?

- Numerous approaches exist
- We picked a technique called “rag”
- RAG is a form of **prompt engineering**

Retrieval-Augmented Generation



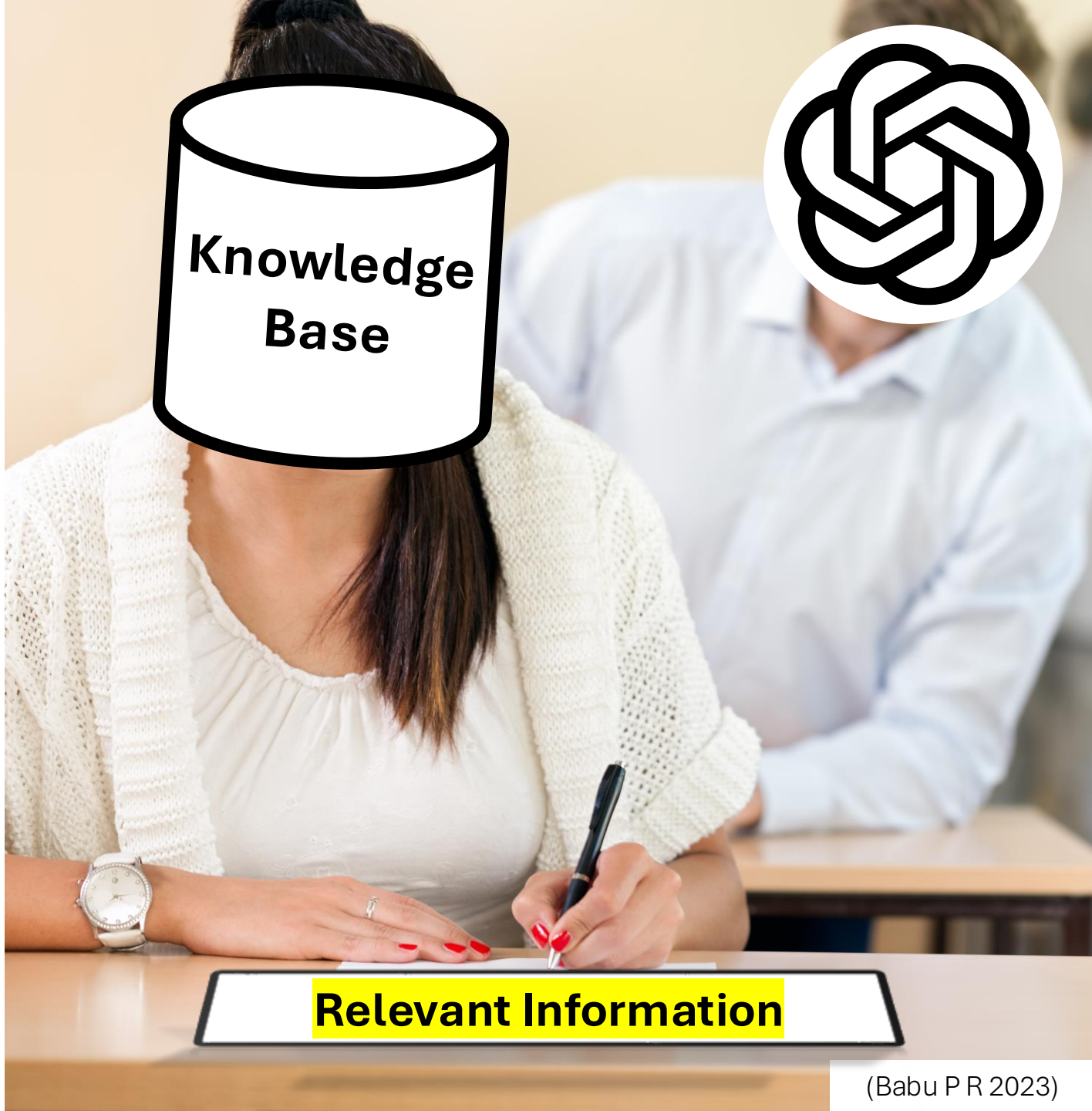
Agenda

- **Introduction to RAG** —— How do we mitigate hallucinations?
- **System Overview** ——— What did we make?
- **System Architecture** —— How does our solution work?
  - Knowledge Base
  - Retrieval Method
  - Query Decomposition
  - Iterative Reasoning
- **Conclusion** ———————— What are the implications?
- **Q & A** ———————————— Ask us about it

Introduction to RAG

What is RAG?

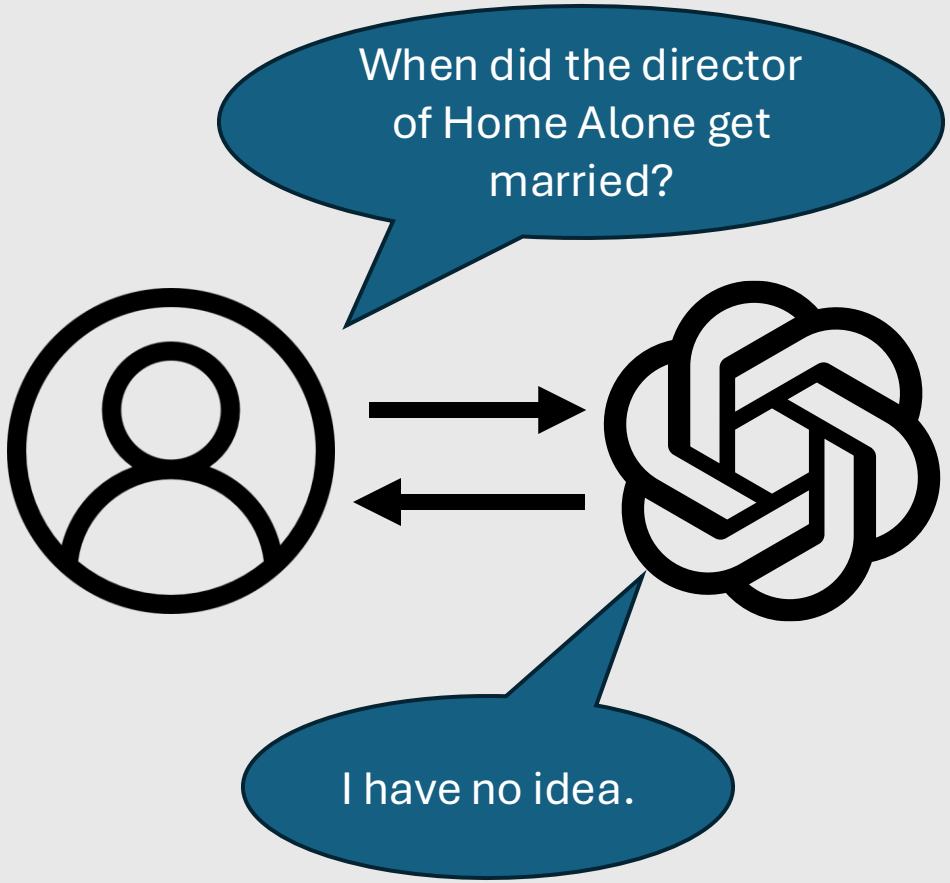
- RAG is a technique to improve LLMs' factual accuracy.
- When an LLM is about to answer a question, we **directly give** it information needed to answer it.
- We store all **answer information** in a **knowledge base**.



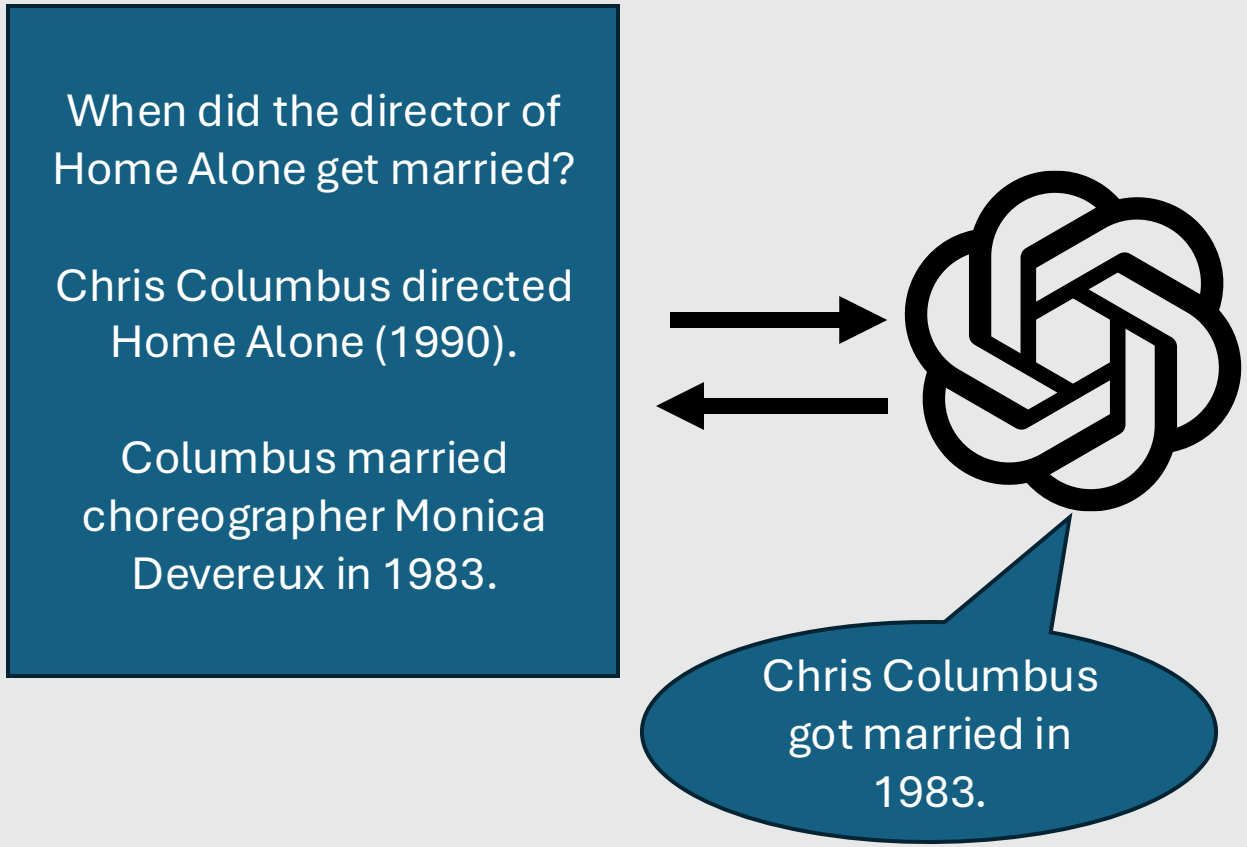
(Babu P R 2023)

Context VS No Context

Prompt without context



Prompt with context



Why did we choose RAG?

RAG allows us to **separate an LLM's knowledge** from the **model**.
This has several benefits:

- **Efficiency**:
  Updating the LLM's knowledge does not require re-training the entire model.
  - This saves a *lot* of time and compute power.

- **Recency:**
  It is easy to update the LLM's knowledge to reflect current world events.

- **Transparency:**
  All the LLM's knowledge is visible instead of hidden in the model weights.
  - This allows LLMs to be easily fact-checked and rectified.

Additionally, RAG has the greatest amount of research and resources for us to reference for our implementation.

How does RAG work?

How do we get information from the knowledge base to answer the user's question?

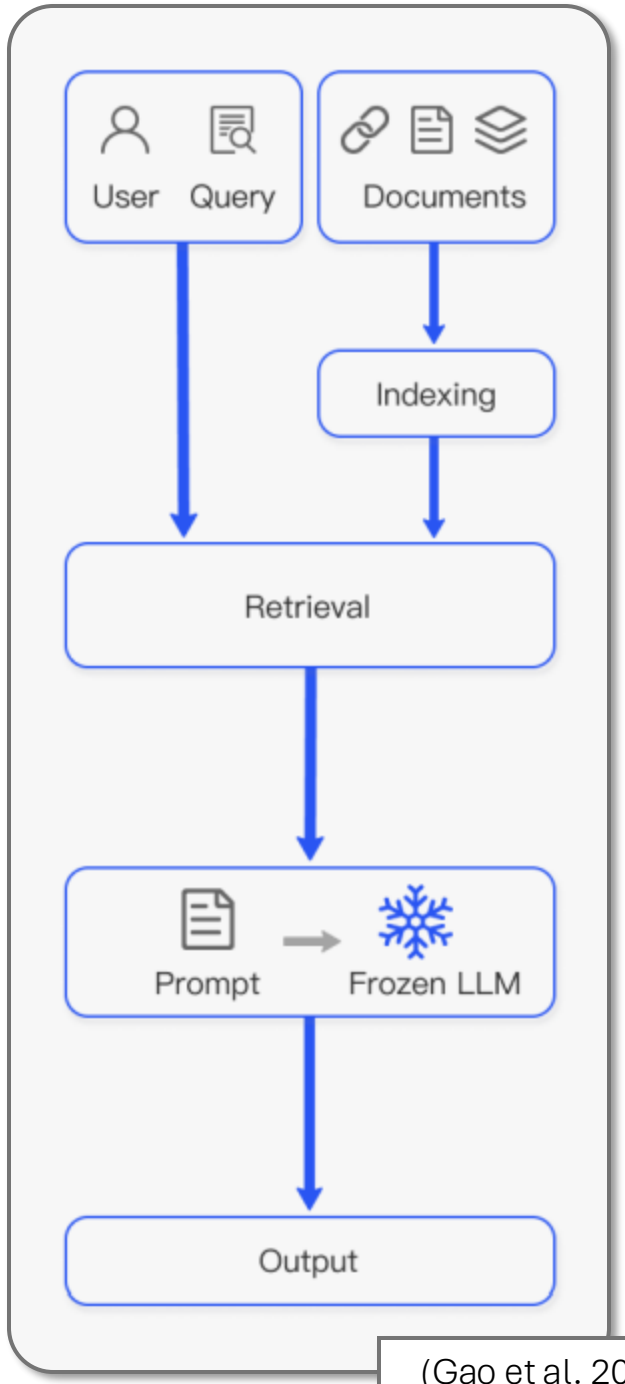
- We obtain it using the **retrieval** process.

How is information stored in the knowledge base?

- We convert text into **embeddings** (vectors).

How can we be sure our model mitigates hallucinations?

- We run **evaluation benchmarks**.



(Gao et al. 2023)

System Overview

The requirements

For our solution, we have the following requirements:

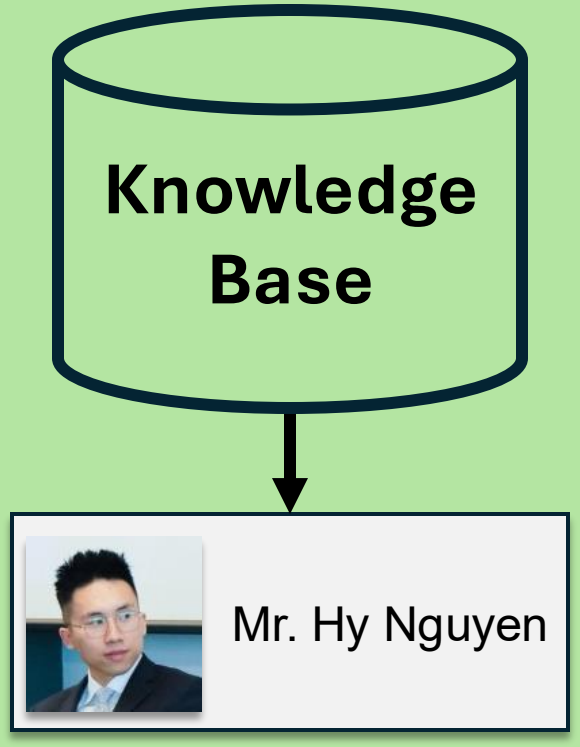
- **Make it from scratch**
  - **No** off-the-shelf RAG libraries.
  - **No** LangChain or Chroma.
- **Only use open source technology.**
  - **No** ChatGPT (property of OpenAI).
- **Make it run on a potato** (our computers).
  - **No** expensive LLM calls or training procedures.

The technique at a glance

The technique at a glance

Knowledge storage

Create an external knowledge base to store all information **(contexts)**.

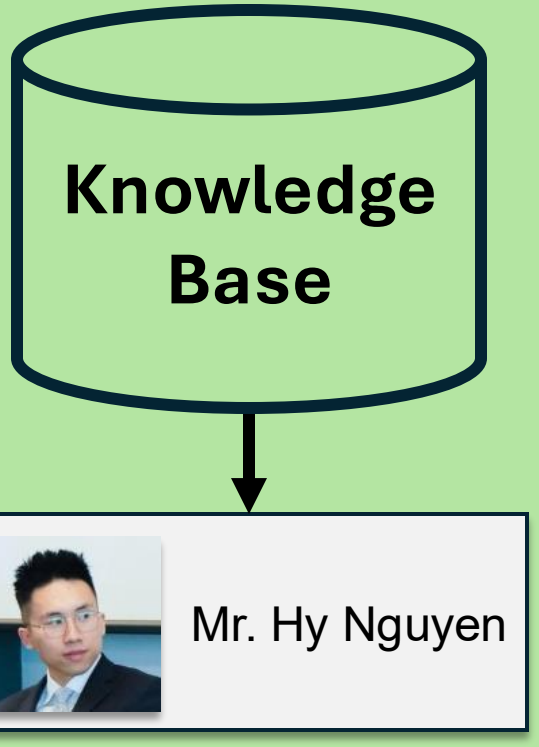


The technique at a glance

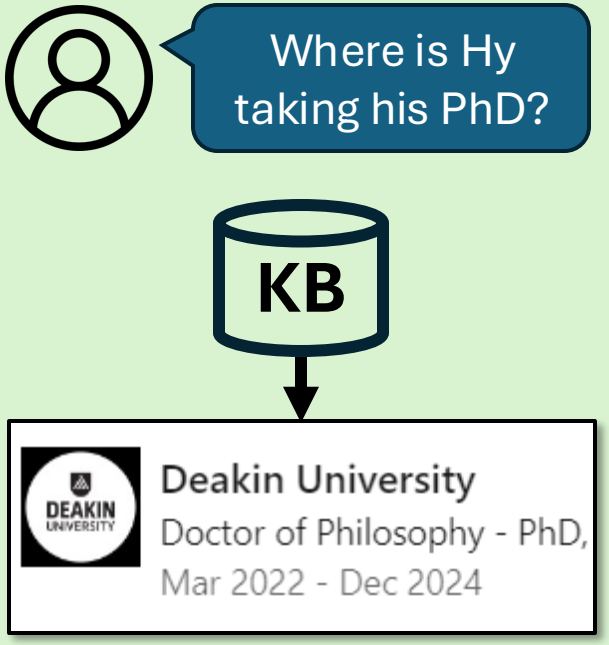
Knowledge storage

Knowledge retrieval

Create an external knowledge base to store all information **(contexts)**.



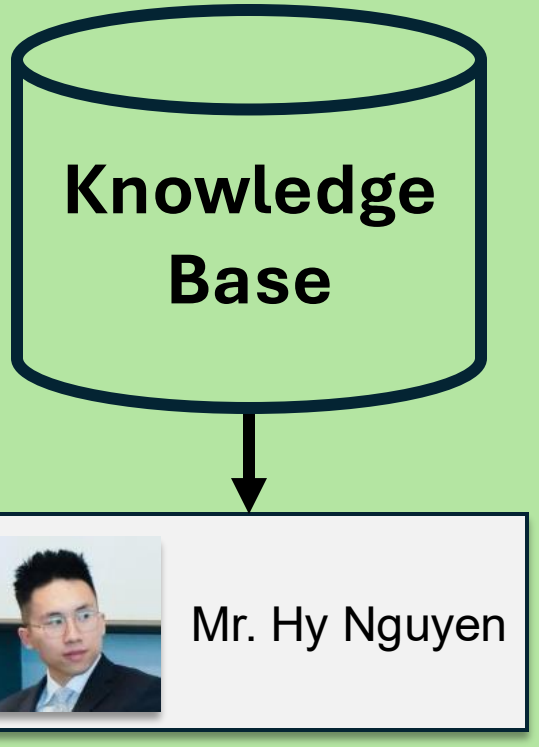
When the user asks a question, find *relevant* contexts to answer it.



The technique at a glance

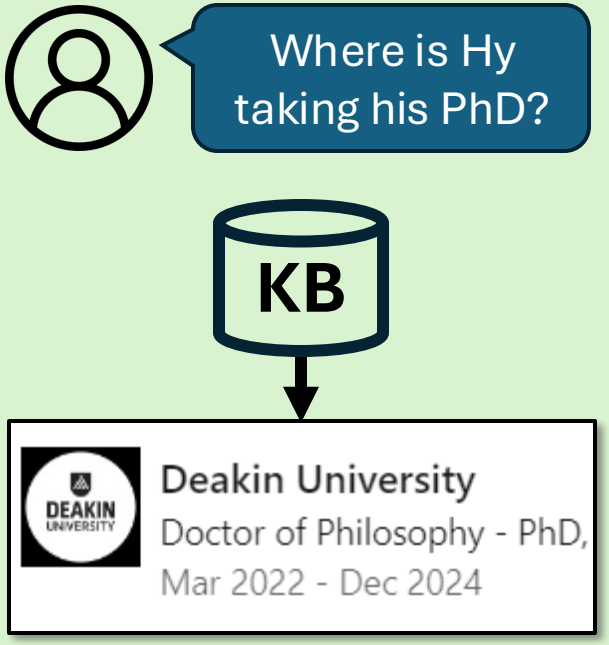
Knowledge storage

Create an external knowledge base to store all information (**contexts**).



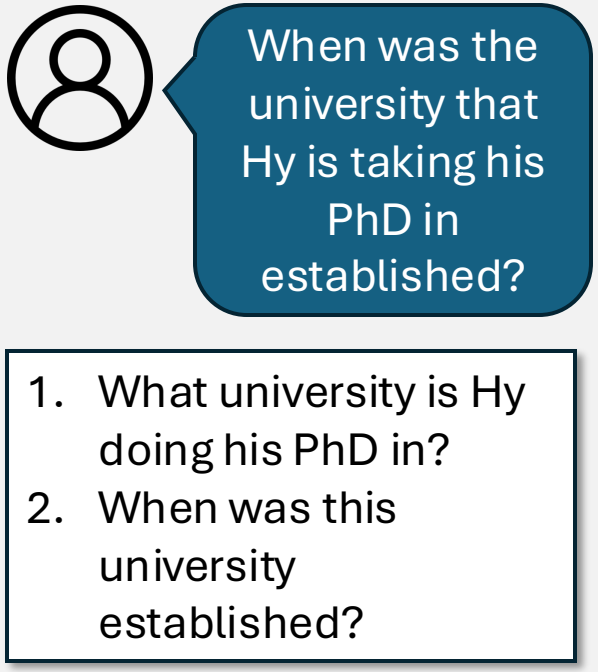
Knowledge retrieval

When the user asks a question, find *relevant* contexts to answer it.



Query decomposition

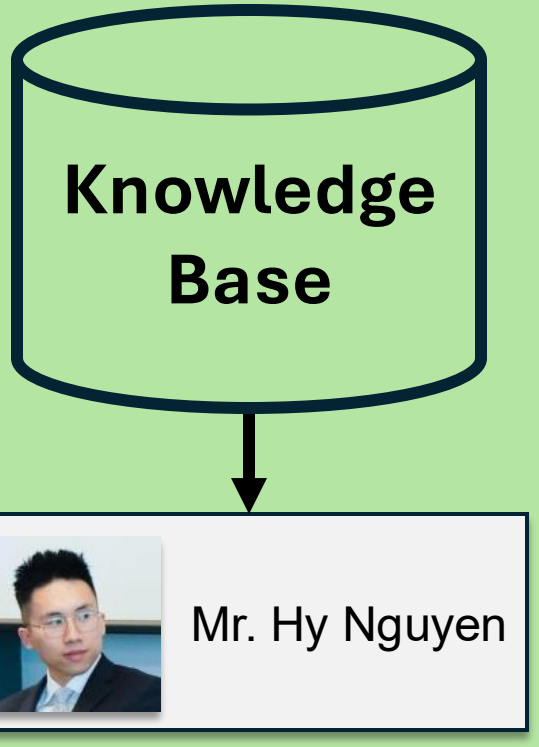
If the user asks a complex question, we must break it down first before getting the context.



The technique at a glance

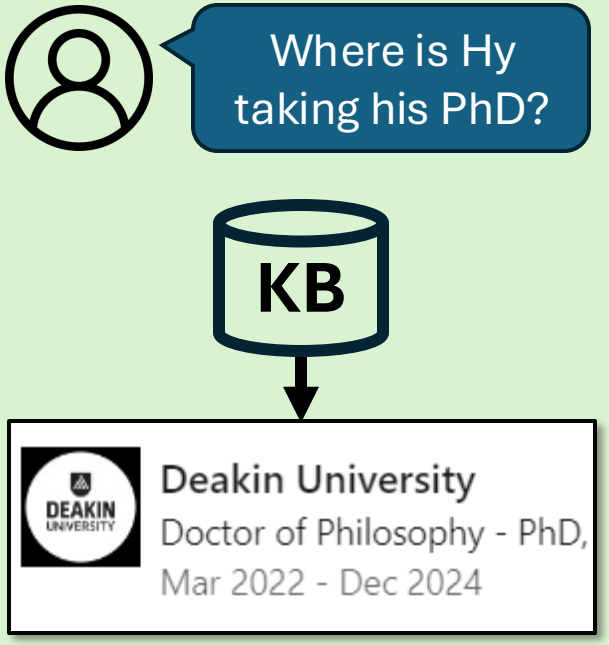
Knowledge storage

Create an external knowledge base to store all information (**contexts**).



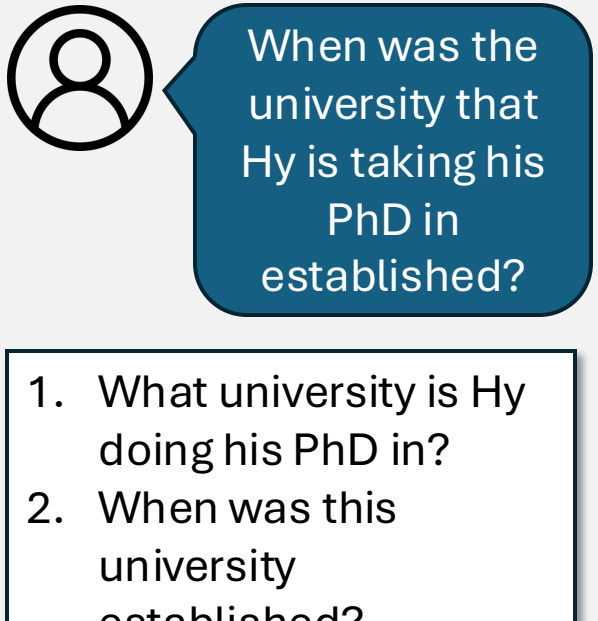
Knowledge retrieval

When the user asks a question, find *relevant* contexts to answer it.



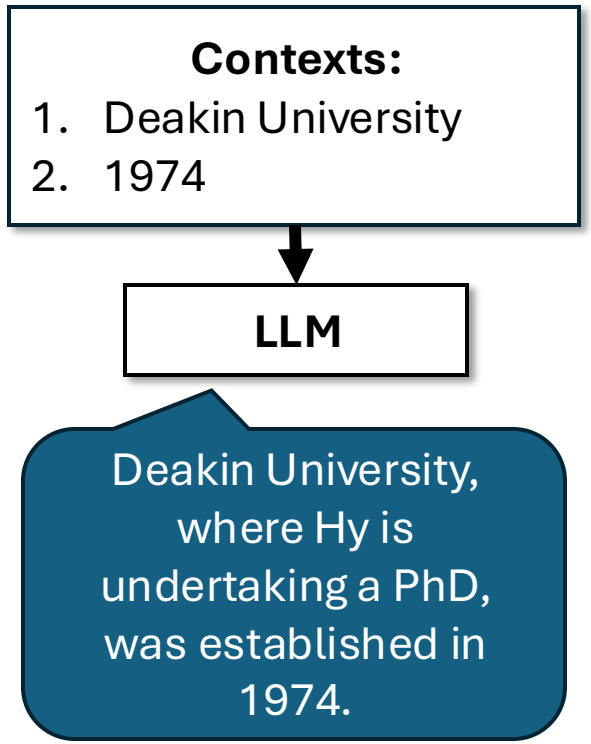
Query decomposition

If the user asks a complex question, we must break it down first before getting the context.



Question answering

Answer the user's question using the context we've retrieved.



System architecture

Knowledge storage

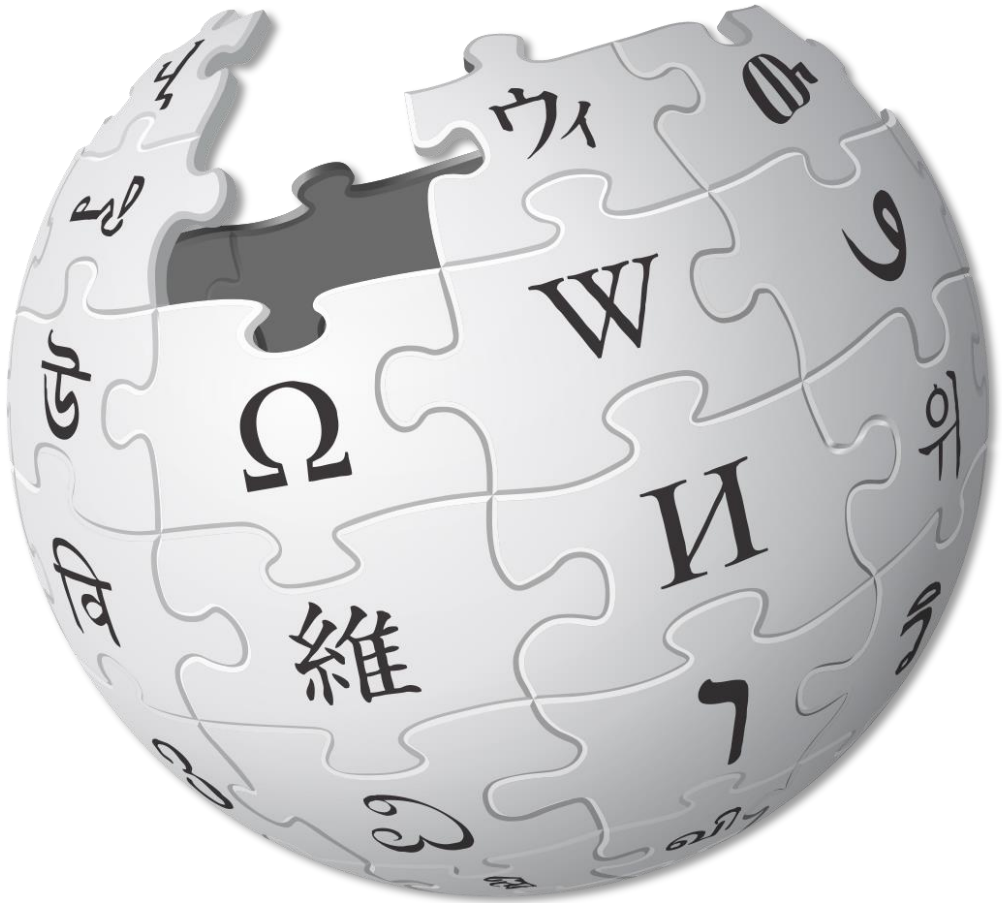
Knowledge base

Knowledge storage | Knowledge retrieval | Query decomposition | Question answering

- **Where do we find knowledge about everything that exists?**

WIKIPEDIA

  - *English Wikipedia specifically…
  - *Doesn’t really cover everything that exists…



Knowledge base: Creation

- **Step 1**: Download **Wikipedia**
  - 33 million articles on English Wikipedia
  - About **21GB** of data
- **Step 2**: Convert it into a Knowledge Base
  - We need a unique data structure

Downloading Wikipedia

Knowledge storage | Knowledge retrieval | Query decomposition | Question answering

We download all English Wikipedia as a .bz2 archive.

enwiki-latest-pages-articles-multistream4.xml-p..>	11-Oct-2024 21:03	865
enwiki-latest-pages-articles-multistream5.xml-p..>	06-Oct-2024 06:04	493080954
enwiki-latest-pages-articles-multistream5.xml-p..>	11-Oct-2024 21:03	865
enwiki-latest-pages-articles-multistream6.xml-p..>	06-Oct-2024 06:04	528066423
enwiki-latest-pages-articles-multistream6.xml-p..>	11-Oct-2024 21:03	868
enwiki-latest-pages-articles-multistream7.xml-p..>	06-Oct-2024 06:05	540724771
enwiki-latest-pages-articles-multistream7.xml-p..>	11-Oct-2024 21:03	871
enwiki-latest-pages-articles-multistream8.xml-p..>	06-Oct-2024 06:05	552779430
enwiki-latest-pages-articles-multistream8.xml-p..>	11-Oct-2024 21:03	871
enwiki-latest-pages-articles-multistream9.xml-p..>	06-Oct-2024 06:05	597528001
enwiki-latest-pages-articles-multistream9.xml-p..>	11-Oct-2024 21:03	871
enwiki-latest-pages-articles.xml.bz2	06-Oct-2024 05:48	22998824471
enwiki-latest-pages-articles.xml.bz2-rss.xml	06-Oct-2024 20:18	781
enwiki-latest-pages-articles1.xml-p1p41242.bz2	06-Oct-2024 04:09	283332977
enwiki-latest-pages-articles1.xml-p1p41242.bz2-..>	06-Oct-2024 20:18	811
enwiki-latest-pages-articles10.xml-p4045403p539..>	06-Oct-2024 04:14	570141958
enwiki-latest-pages-articles10.xml-p4045403p539..>	06-Oct-2024 20:18	838
enwiki-latest-pages-articles11.xml-p5399367p689..>	06-Oct-2024 04:14	548417164
enwiki-latest-pages-articles11.xml-p5399367p689..>	06-Oct-2024 20:18	838
enwiki-latest-pages-articles11.xml-p6899367p705..>	06-Oct-2024 04:05	52279530
enwiki-latest-pages-articles11.xml-p6899367p705..>	06-Oct-2024 20:18	838
enwiki-latest-pages-articles12.xml-p7054860p855..>	06-Oct-2024 04:13	452965013
enwiki-latest-pages-articles12.xml-p7054860p855..>	06-Oct-2024 20:18	838
enwiki-latest-pages-articles12.xml-p8554860p917..>	06-Oct-2024 04:08	182795719
enwiki-latest-pages-articles12.xml-p8554860p917..>	06-Oct-2024 20:18	838
enwiki-latest-pages-articles13.xml-p10672789p11..>	06-Oct-2024 04:09	255597770
enwiki-latest-pages-articles13.xml-p10672789p11..>	06-Oct-2024 20:18	844

Extract raw text

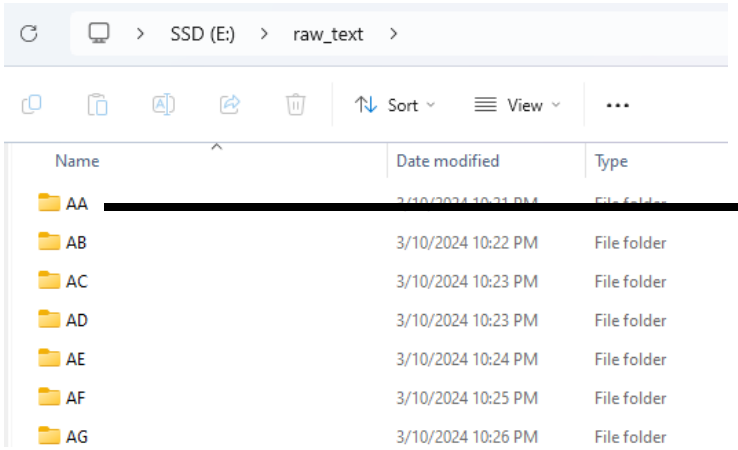
enwiki-latest-pages-articles.xml.bz2

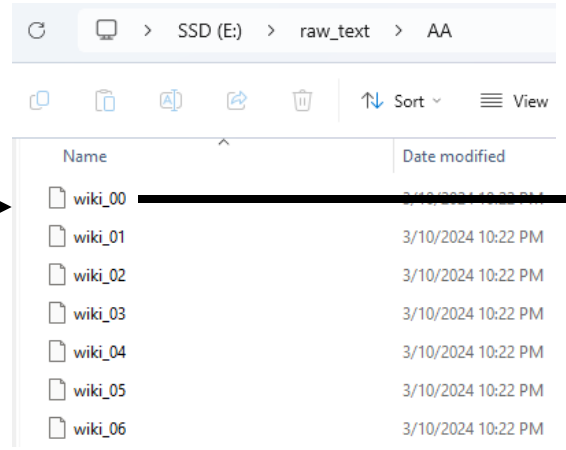
21 GB

→ WikiExtractor →

Raw text

- **All articles stored in a series of text files.**
  - Each article = JSON object.

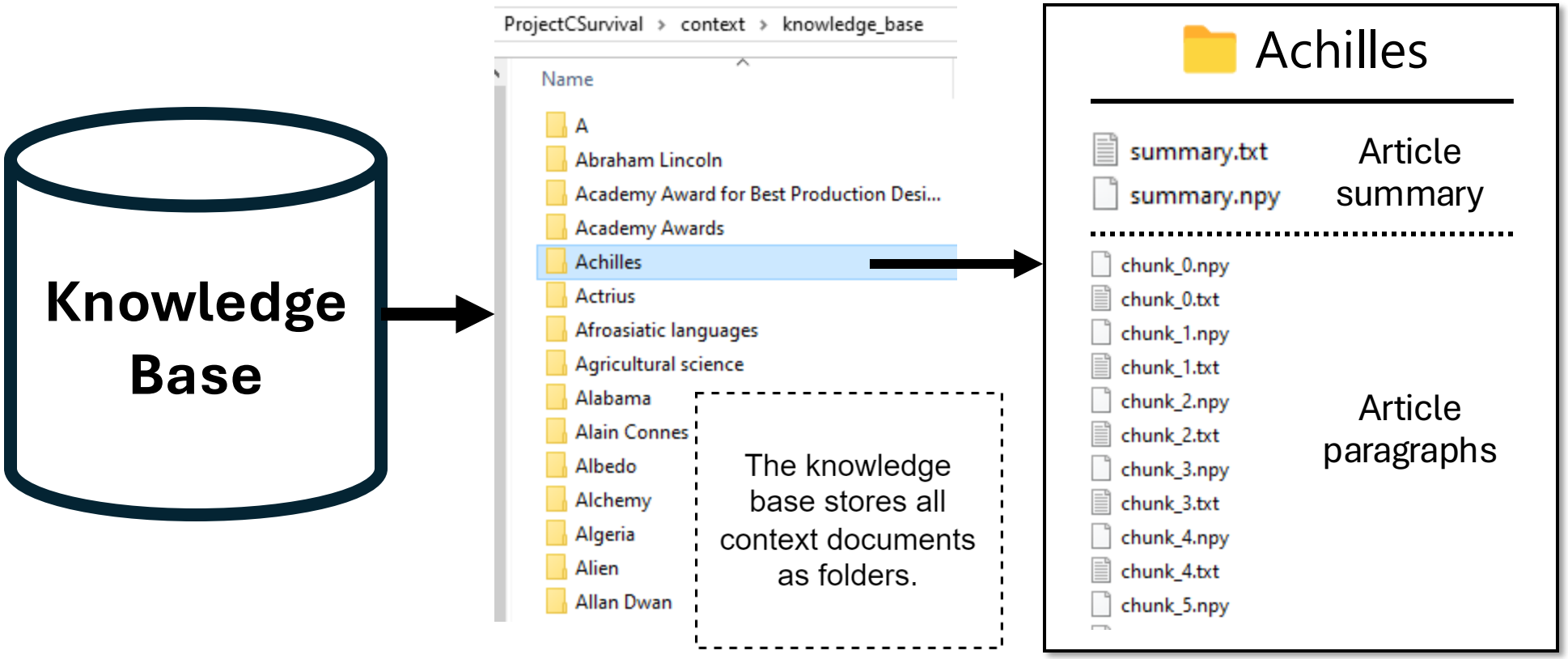




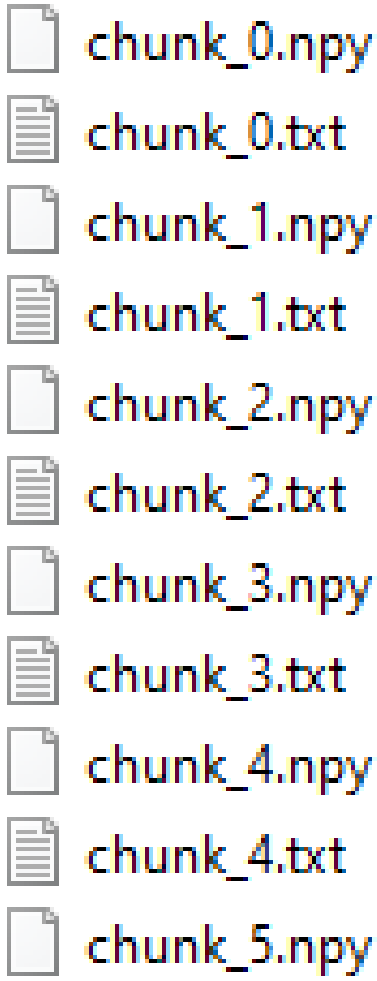
{"id": "12", "url": "https://en.wikipedia.org/wiki?curid=12", "title": "Anarchism", "text": "Anarchism\n\nAnarchism is a political philosophy and movement that is against all forms of authority and seeks to abolish the institutions it claims maintain unnecessary coercion and hierarchy, typically including the state and capitalism. Anarchism advocates for the replacement of the state with stateless societies and voluntary free associations. As a historically left-wing movement, this reading of anarchism is placed on the farthest left of the political spectrum, usually described as the libertarian wing of the socialist movement (libertarian socialism).\n\nAlthough traces of anarchist ideas are found all throughout

Knowledge base

- We must convert the Wikipedia raw text into a **knowledge base** for it to be usable.



Storage of text information



All textual information is stored in two forms:

Raw Text: As raw text in a UTF-8 encoded **.txt** file

Vector: As an embedding NumPy array **.npy** file.

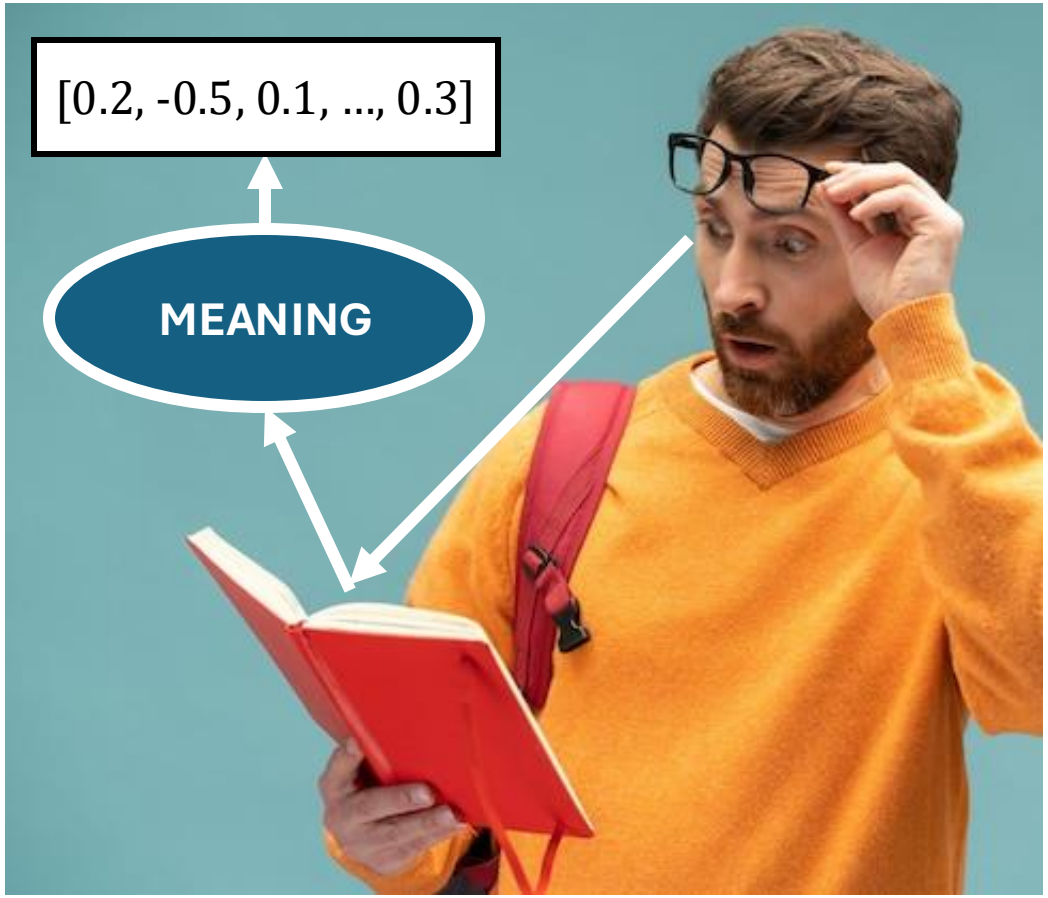
- **What is a text embedding?**

What is a text embedding?

A text embedding mimics how humans understand written information in a way computers can understand.

Specifically, it is a **numerical representation** of **text** which:

- captures the **semantic meaning** of the text
- as a vector of N dimensions
- within a vector space where texts with *similar meanings* are *near each other*.



https://www.freepik.com/premium-photo/shocking-facts-astonished-smart-bearded-student-man-reading-book-with-surprised-expression-amazed-by-story-being-shocked-by-interesting-information-indoor-studio-shot-isolated-blue-background_27394583.htm

(James Jie Pan, Wang & Li 2024, Cao 2024)

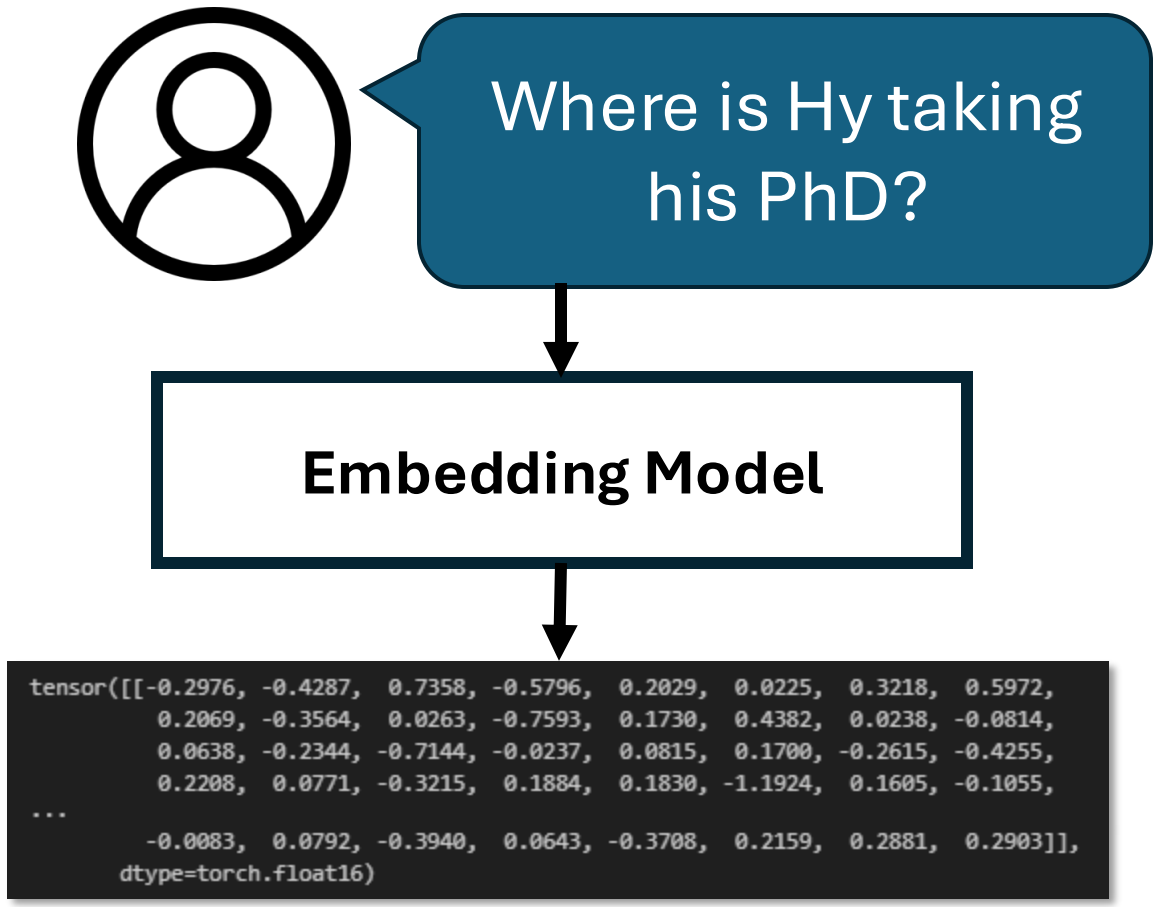
Universal text embeddings

The latest advancement in text embedding models is **Universal Text Embedding.**

Universal Text Embedding models aim to:

- generate *general-purpose representations* of text that is
- *usable for various NLP tasks and domains*,
- regardless of the text's length, domain, task, or language.

Intuition:
Humans' understanding of textual information is *not task-specific*.



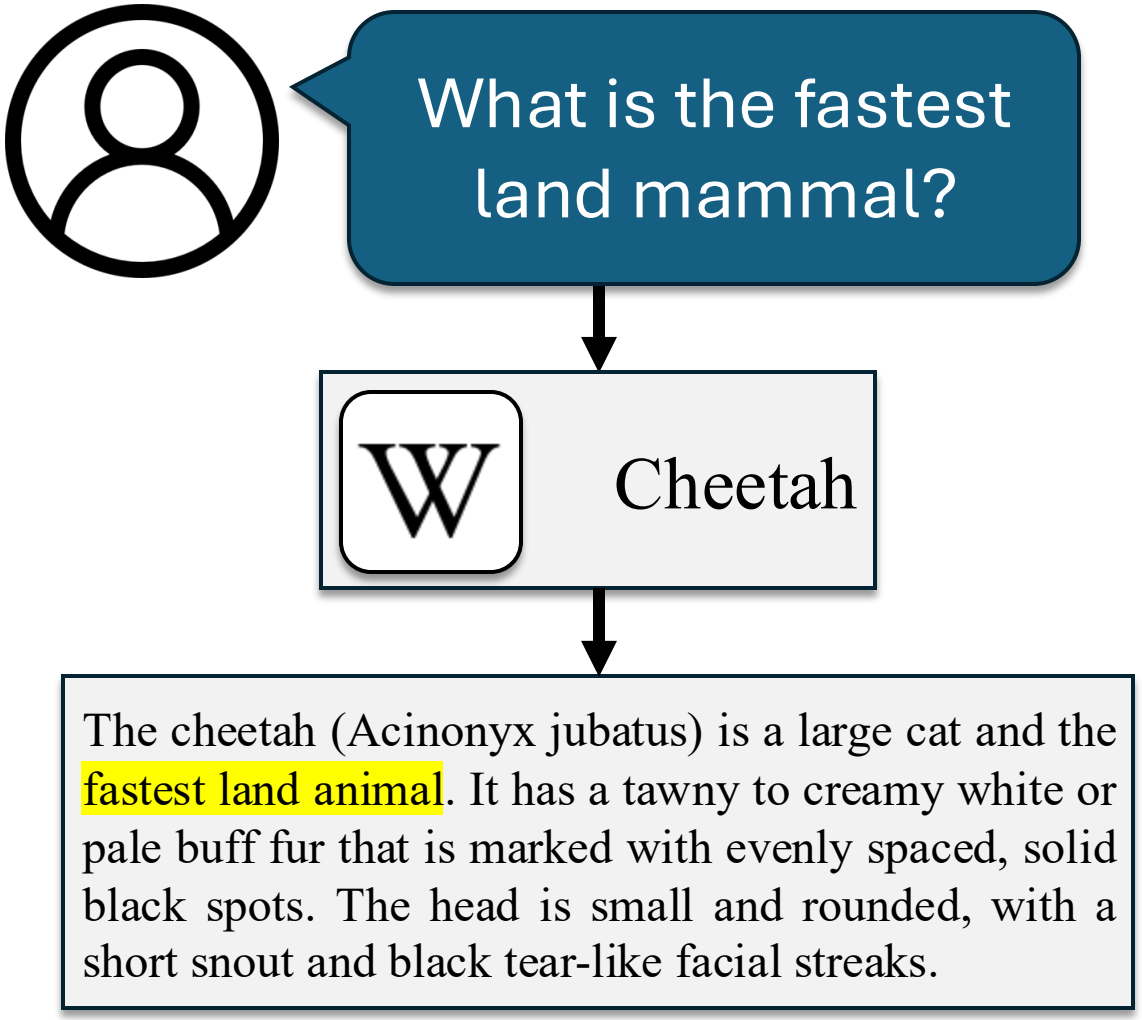
(Cao 2024, Conneau & Kiela 2018)

System architecture

Knowledge retrieval

Retrieval

- When the user asks a **question**, we must get the answer from the **knowledge base**, if it is obtainable.
- To do this, we must find a **paragraph** from a **Wikipedia article** which provides the **answer** to the user's question.

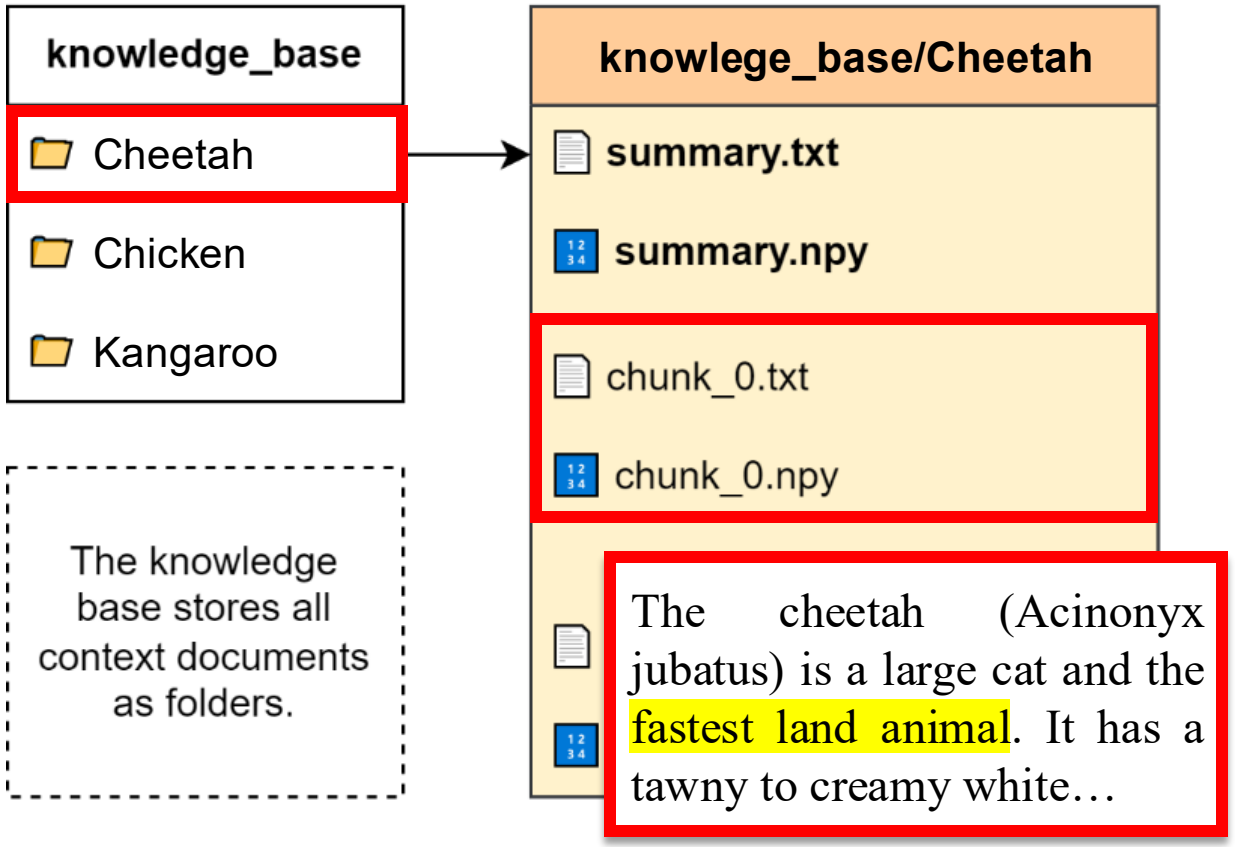


Retrieval

- We use **two steps** to find the answer for a user's question.

1. **Find a Wikipedia article** whose summary is most similar to the question.
2. **Find a paragraph (chunk)** from the article which is most similar to the question.

- *How do we know* which article and paragraph to select to answer the user's question?



Retrieval methods

If we have a **question**, how can we find the **answer** from a library of books?

1) Find the book with the **most word matches** to the question?
2) Read every book and pick the most relevant one by **meaning**?
3) **Skip books we know aren't relevant** and only search relevant ones?

Retrieval methods

1) Find the book which has the most exact word matches to the question

- Search for every detail (**word match**) of the book to our need (**query**). Similarly to BM25 Algorithm
- Calculate the **rate of term** with **frequency of term** in respect of the **length of the document.**

$$score\ (D,Q) = \sum_{i=1}^{n} \left(\ln\left(\frac{N - n(q_i) + 0.5}{n(q_i) + 0.5} + 1 \right) \cdot \left(\frac{f(q_i, D).(k_1 + 1)}{f(q_i, D) + k_1(1 - b + b \cdot \frac{|D|}{avgdl})} \right) \right)$$

Total number of documents

Number of documents contain term q_i

Frequency of term q_i appear in document

Number of documents contain term q_i

Frequency of term q_i appear in document

Document length /w respect to Avg. total docs length

$$score\ (D,Q) = Summary\left[\left(\frac{number\ of\ doc\ don't\ have\ word}{number\ of\ doc\ have\ word} \right) \cdot \left(\frac{freq\ of\ word\ appear\ in\ Doc}{freq\ of\ word + Doc\ length\ /w\ respect\ to\ (avg\ Docs)} \right) \right]$$

⇒ **Sparse Retrieval**

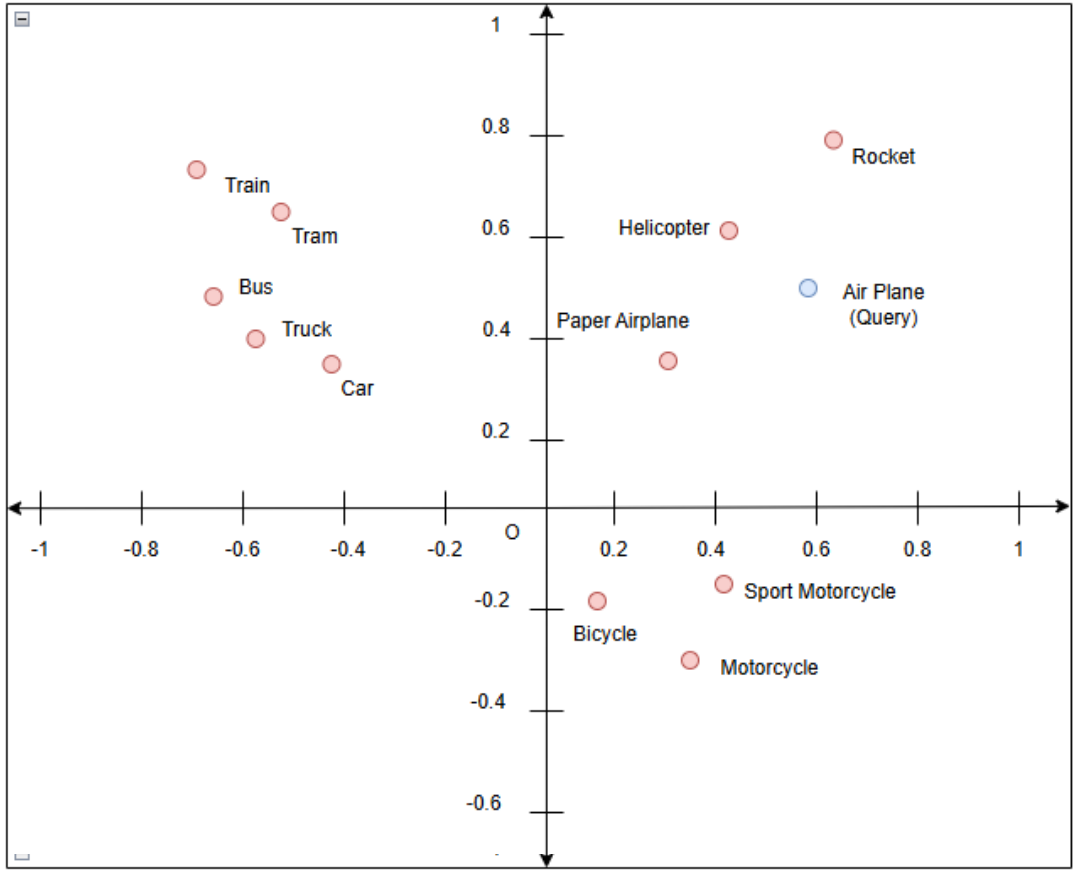
$$score(D,Q) = Sumary[(how\ rare\ the\ term\ is) \cdot (frequency\ of\ word\ to\ doc\ length)]$$

Retrieval methods

2) Read every book and pick the most relevant one by meaning

- Use **vector embeddings** to measure similarity between each **article** to the query.
- Common method for measuring:
  - Euclidean Distance Similarity
  - Cosine Similarity
  - Dot Product Similarity

⇒ **Dense Retrieval**



Dense retrieval

Euclidean Distance Similarity

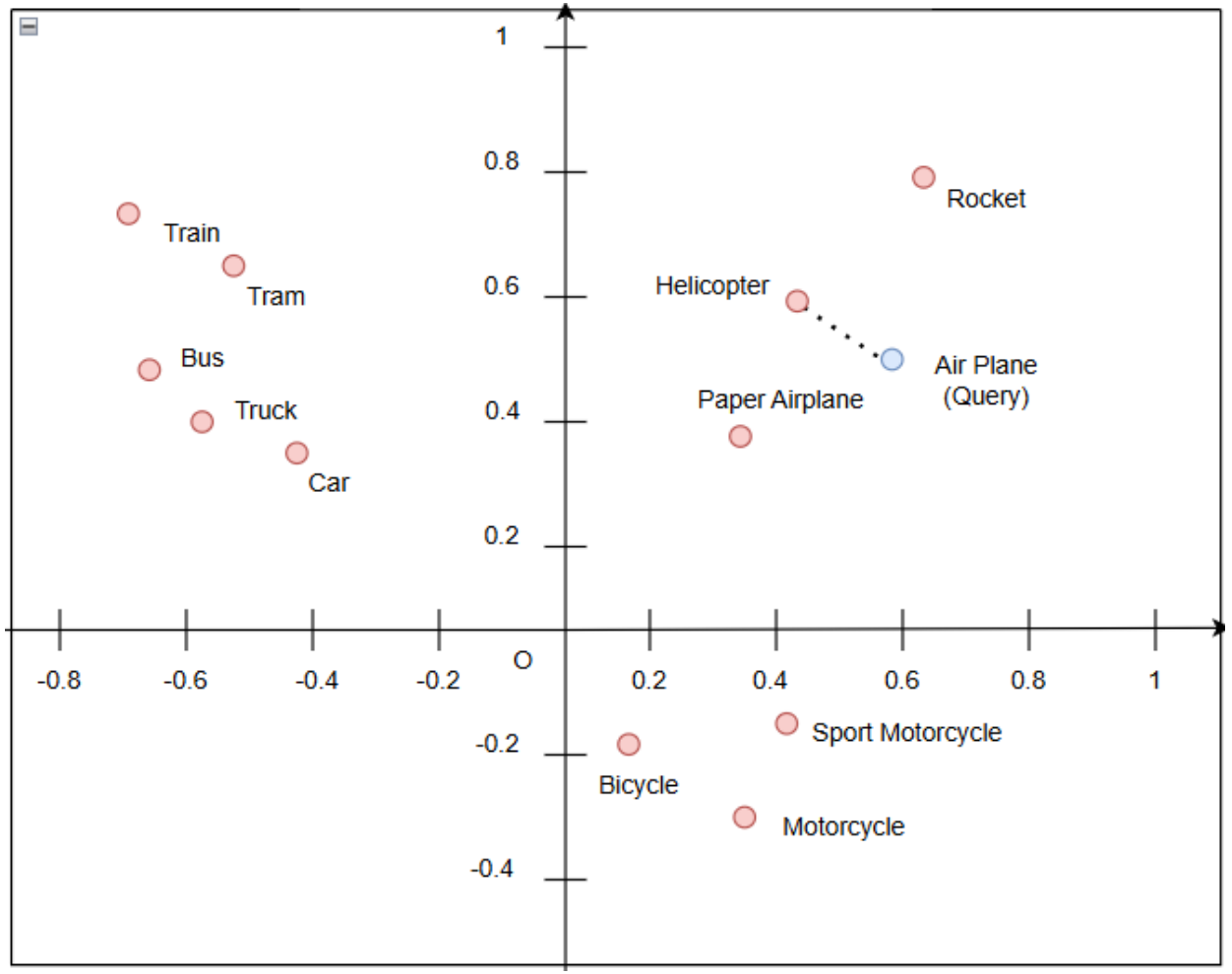
- Based on calculation of the **distance between two points** in a multi-dimensional space.
- The **shorter the distance** between two points, the **greater the similarity** between them.

Euclidean Distance:

$$d(p,q) = \sqrt{(p_1 - q_1)^2 + (p_2 + q_2)^2 + \cdots + (p_n - q_n)^2}$$

***p,q*:** vector point

***n*:** total dimension of the space



Dense retrieval

Cosine Similarity

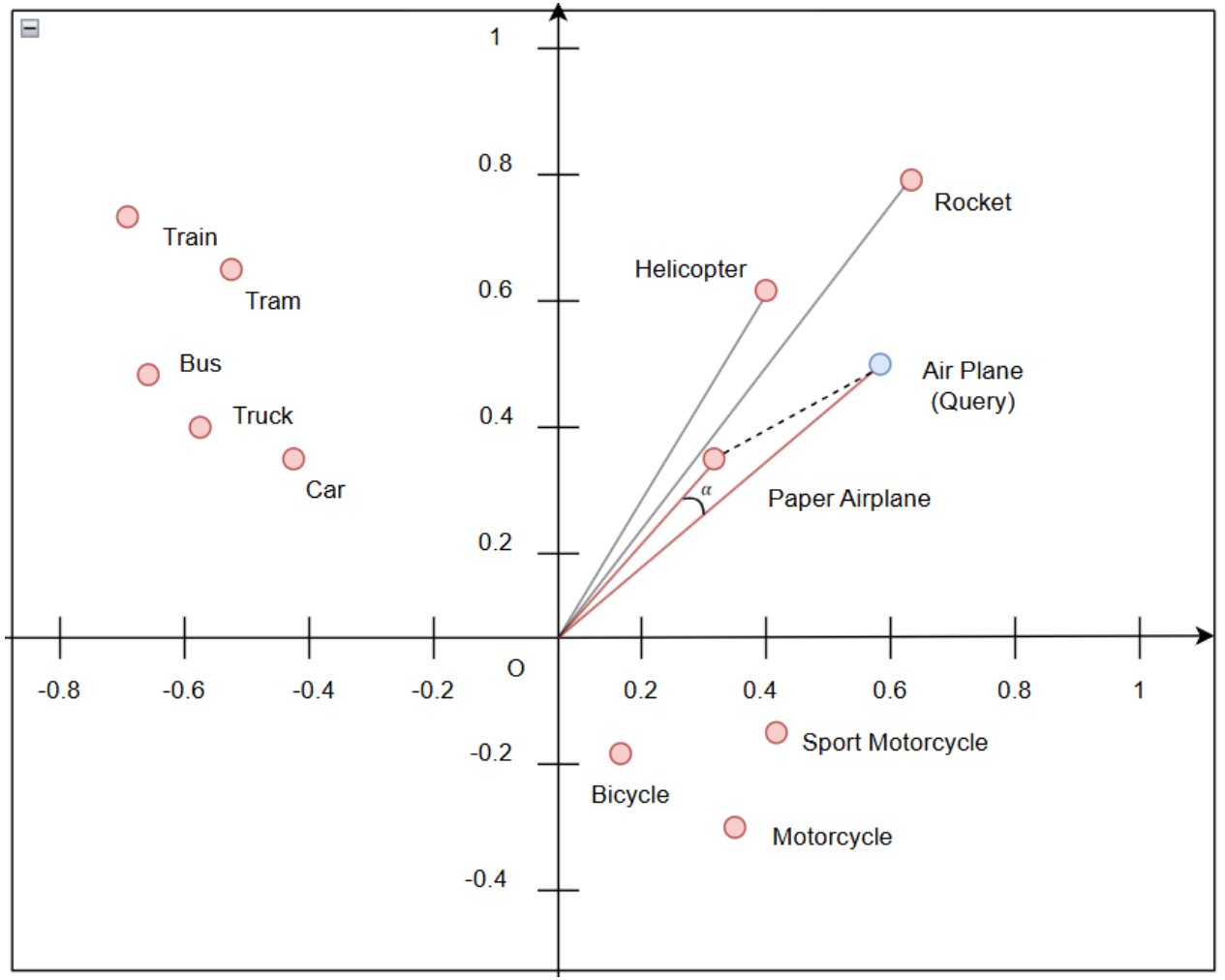
- Based on calculation the **angle between two vectors** in space.
- The **smaller the angle**, the **greater the similarity** between them.

Cosine Similarity:

$$cosine(\alpha) = \frac{\vec{a}.\vec{b}}{\|a\|.\|b\|}$$

$\vec{a}.\vec{b}$: vector point

α: angle between two vectors



Dense retrieval

Dot Product

- Based on calculation the dot product, which involves the **lengths of two vectors** in a multi-dimensional space.
- A **higher dot product value** indicates **greater similarity** between the vectors.

Dot Product

$$p \cdot q = p_1 \times q_1 + p_2 \times q_2 + \cdots + p_n \times q_n$$

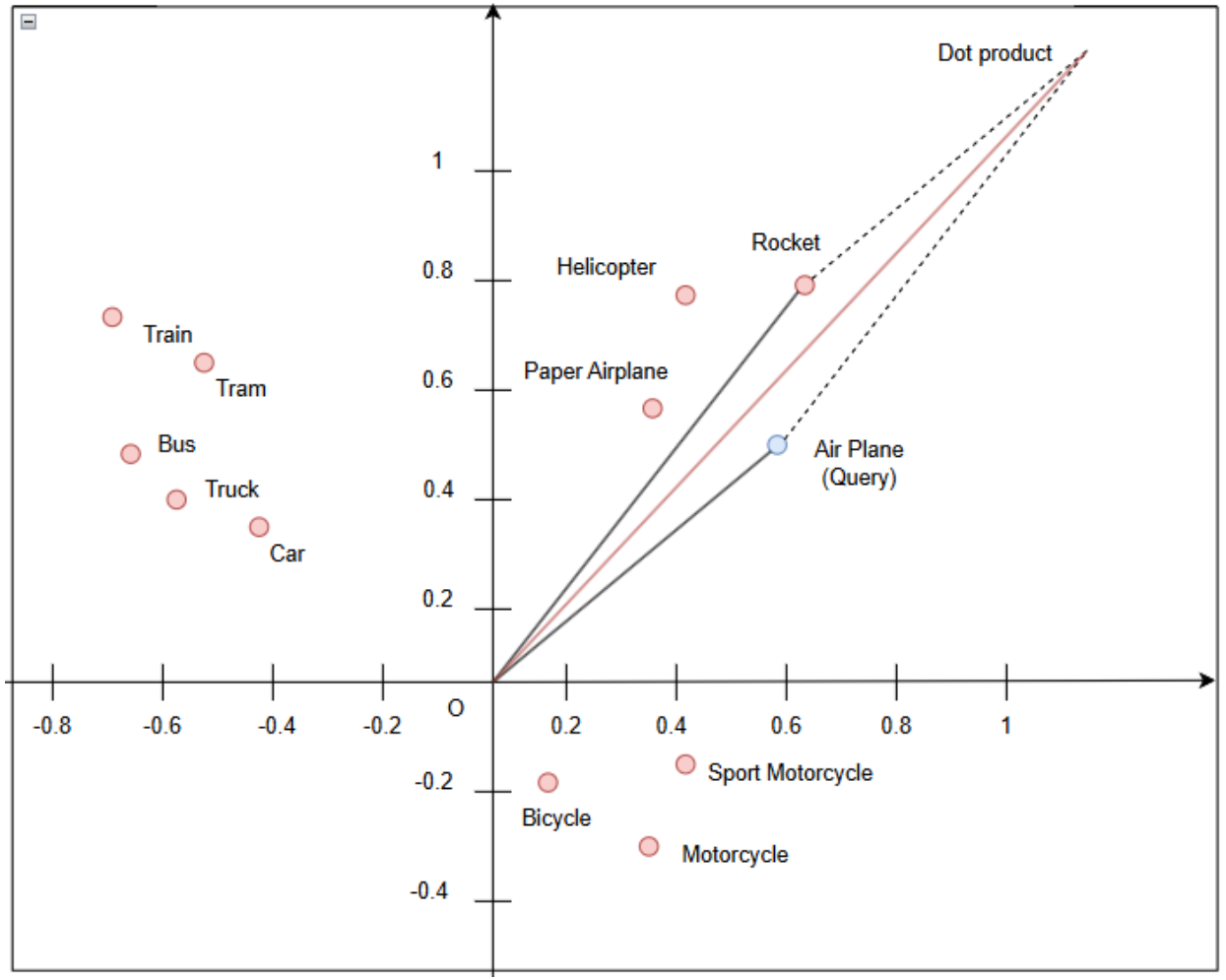
Or

$$\vec{a} \cdot \vec{b} = \|a\|.\|b\|.cosine(\alpha)$$

p,q / $\vec{a}.\vec{b}$: vector

α: angle between two vectors

n: total dimension of the space



Dense retrieval

Find a Wikipedia article that has the most similar meaning to the user's question.

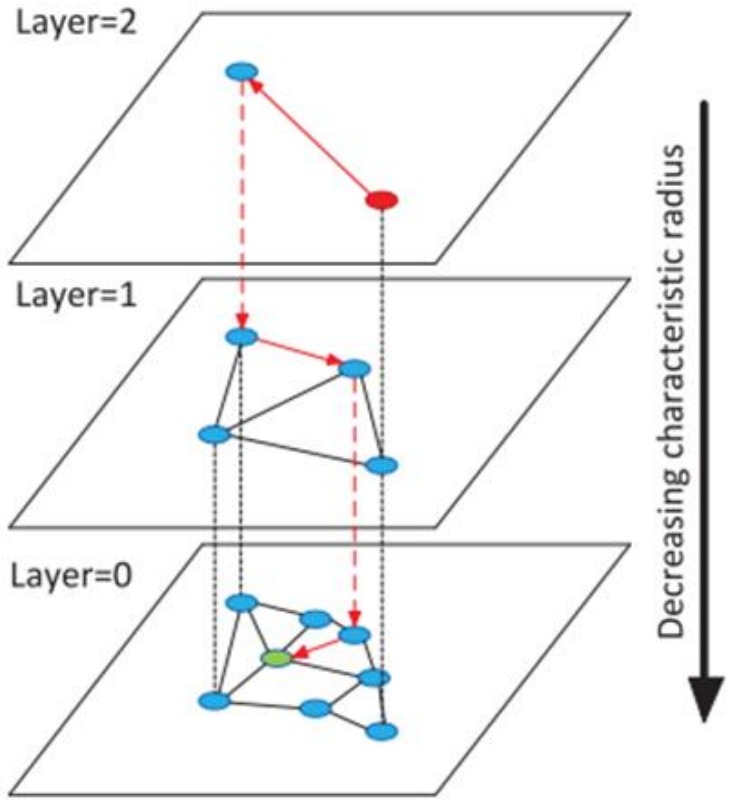
- Calculate similarity of **every vector** in the vector database and **compare** it with the **query vector**.
- It is **not a cost-effective** solution.
- My group also call it as "**Exhaustive Retrieval**"

Retrieval methods

3) Skip books we know aren't relevant and only search relevant ones

- Search for meaning but skip the irrelevant ones
- Similar to algorithm Hierarchical Navigable Small World (HNSW).
- Hierarchical graph structure to efficiently navigate and retrieve nearest neighbors

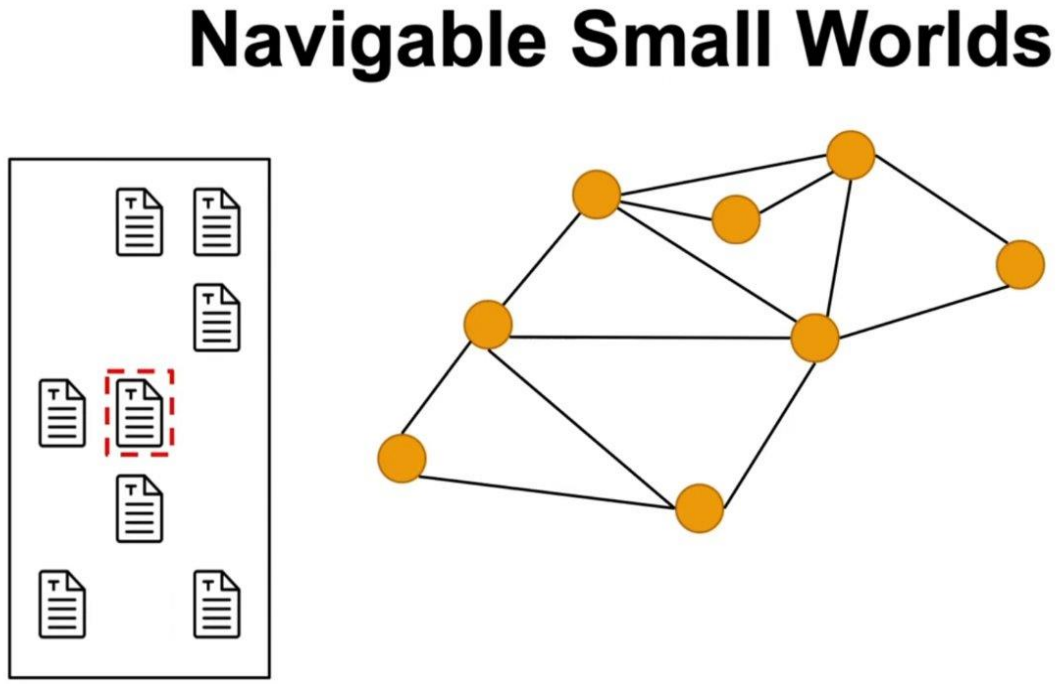
⇒ Hierarchical Retrieval



Navigable Small Worlds (NSW)

Navigable Small World (NSW)

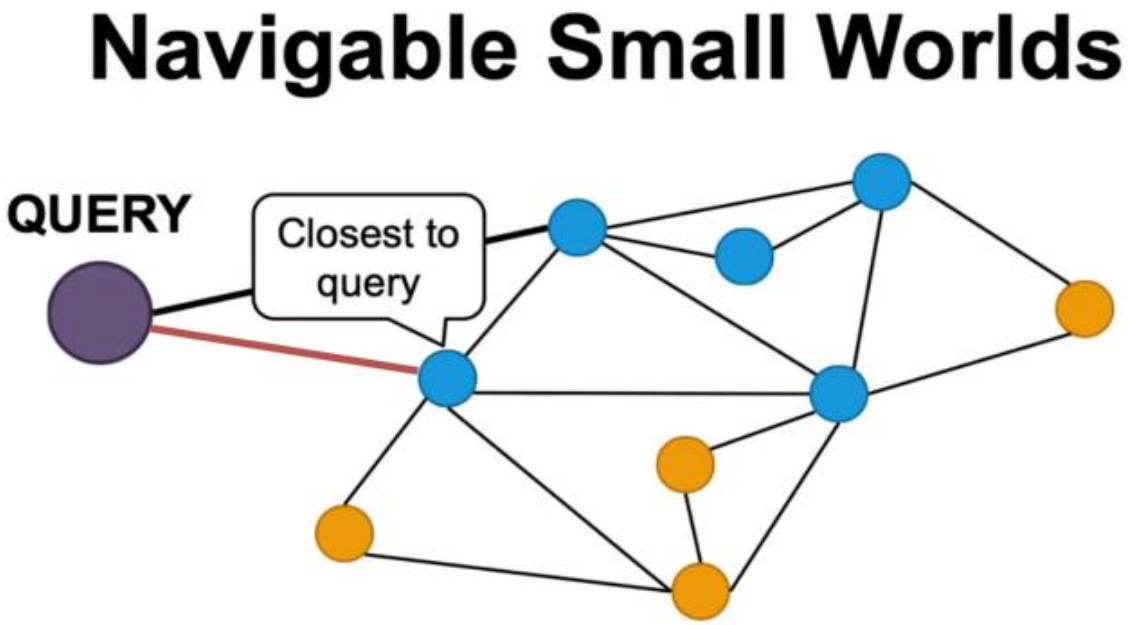
- Store each **embedding chunked of paragraph/document** as a **node**.
- Each node **connect** to **M closest neighbors**



Navigable Small Worlds (NSW)

Navigable Small World (NSW)

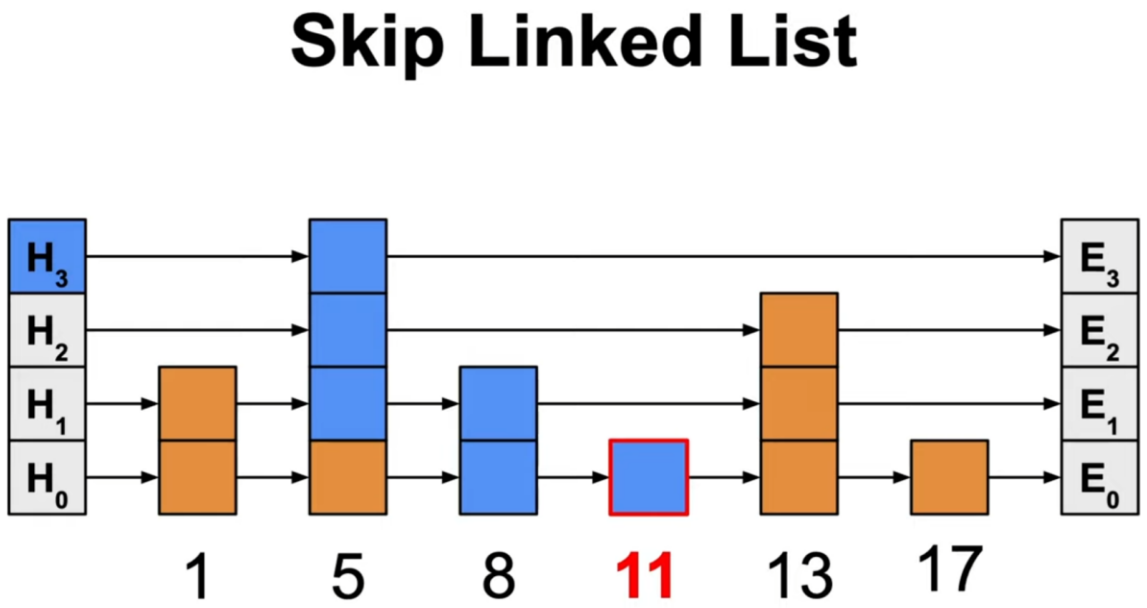
- Store each **embedding chunked of paragraph/document** as a **node**.
- Each node **connect** to **M closest neighbors**
- Using **distance metric** (Euclidean, Manhattan distance,...) to determine which is **closest node to the query**.



Skip Linked List

Skip Linked List

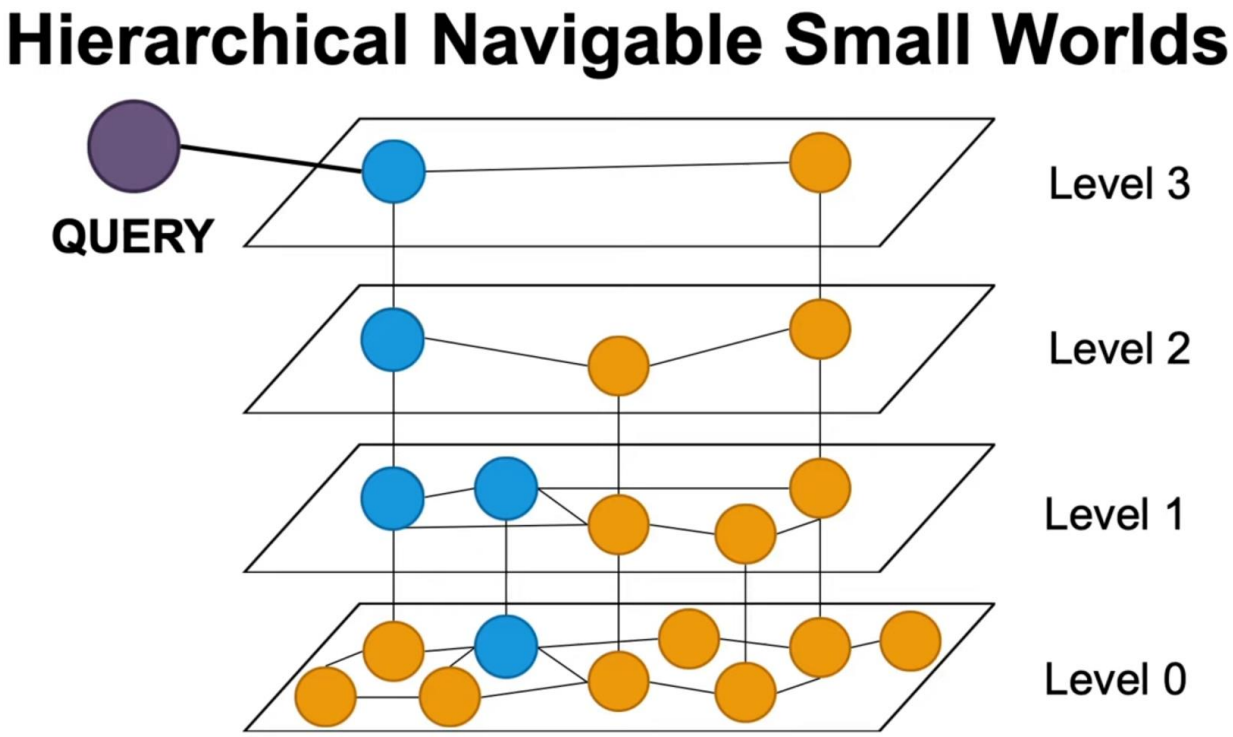
- Start with a random number node at the top layer, then process to the bottom layer with all nodes.
- We have a list of number, and we want to find target number "11"
  - We random choose a number "5" and compare it with target "11"
  - We skip number < "5" and navigate to the number in range of [5,13]
  - Repeat the process until we find 11



Skip Linked List

Hierarchical Navigable Small Worlds (HNSW)

- Combine NSW and Skip Linked List
- HNSW Construction (*efConstruction*):
  - Search through each set of nodes (*ef*)
  - Choose the most similarity then pass to next layer
  - Repeat the process until we find the most similarity:
    - All nodes searched
    - Base layer (Layer 0)
    - Distance(next_closest) > Distance(furthest)



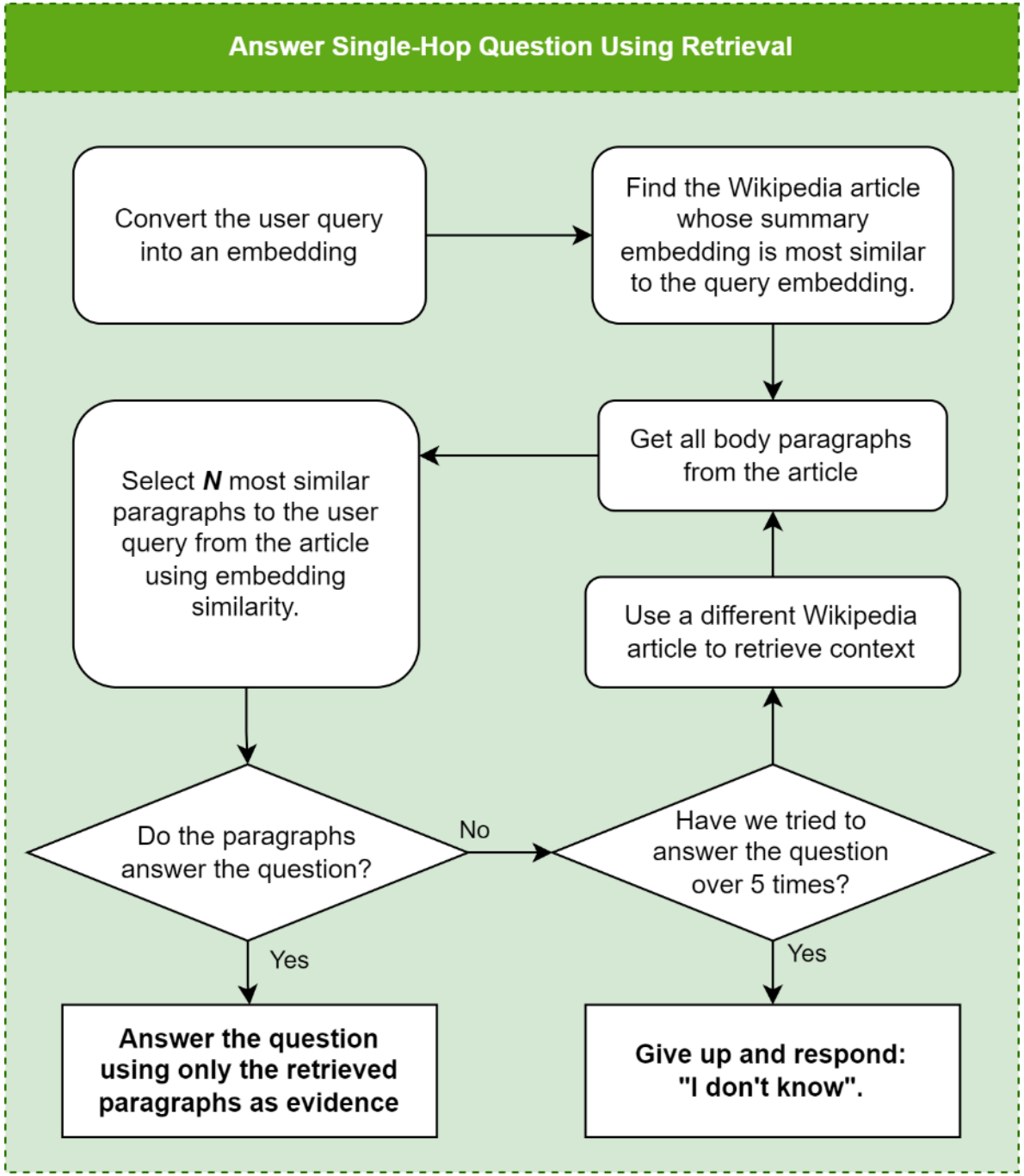
Retrieval methods

Methods:

- Sparse Retrieval (**BM25**)
  - Uses term **frequency and inverse document frequency to rank documents** based on relevance, making it effective for **traditional keyword-based search**.
- Dense Retrieval (**Cosine Similarity**)
  - Represents q**ueries and documents as dense vectors, calculating similarity** to find the best matches, suitable for **semantic search**.
- Hierarchical Retrieval (**Graph layer structure**)
  - Utilizes a **hierarchical graph structure to efficiently navigate and retrieve nearest neighbors**, offering **faster semantic search** in large vector spaces.

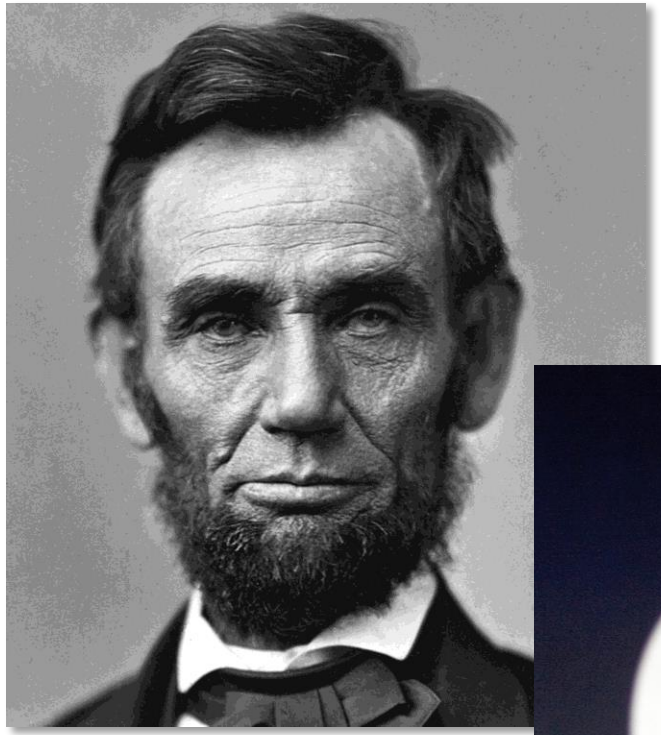
System architecture

Query decomposition


Answer Single-Hop Question Using Retrieval
Convert the user query into an embedding
Find the Wikipedia article whose summary embedding is most similar to the query embedding.
Get all body paragraphs from the article
Select N most similar paragraphs to the user query from the article using embedding similarity.
Use a different Wikipedia article to retrieve context
Do the paragraphs answer the question?
No
Have we tried to answer the question over 5 times?
Yes
Yes
Answer the question using only the retrieved paragraphs as evidence
Give up and respond: "I don't know".

Query decomposition

- "How many years elapsed between Lincoln's presidency and the release of the iPhone?"
- We *can't* answer this question with one Wikipedia article!
- **We need to break it down into two questions.**
  1. When was Abraham Lincoln president?
  2. When was the iPhone released?

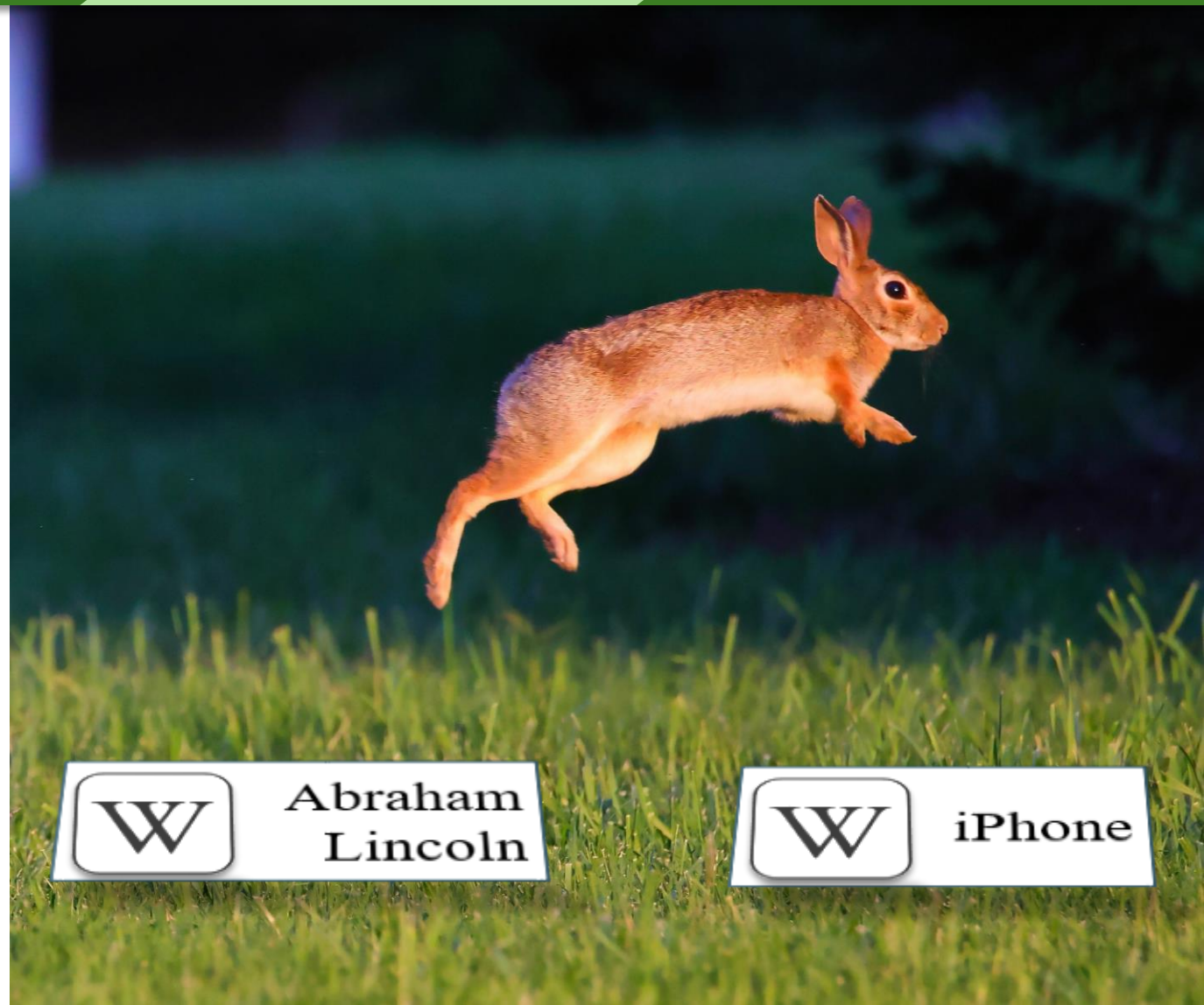


Query decomposition

There are two types of questions:

- **Single-hop**
  - Only one article is needed to determine answer.
    - "Who is the CEO of Facebook?"
    - → Facebook CEOs
    - →Mark Zuckerburg
- **Multi-hop**
  - Multiple articles are needed to find an answer.
  - "How the King of Soccer die?"
    - Who is the King of Soccer? → **Pelé** (Edson Arantes do Nascimento)
    - How did Pelé die? → **Colon cancer**

(Vu et al. 2023)



Query decomposition

How do we automatically break complex questions down into simpler questions?

- **We can use a Llama.**
- **Specifically, Meta Llama 3.1, an open source Large Language Model.**
- We *ask* Llama to break a complex question into simple ones using a **prompt**.

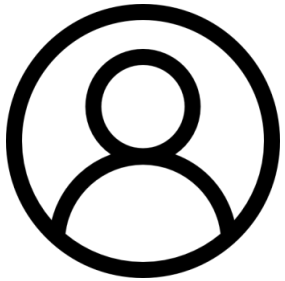
Query decomposition

Question Decomposition Specialist:

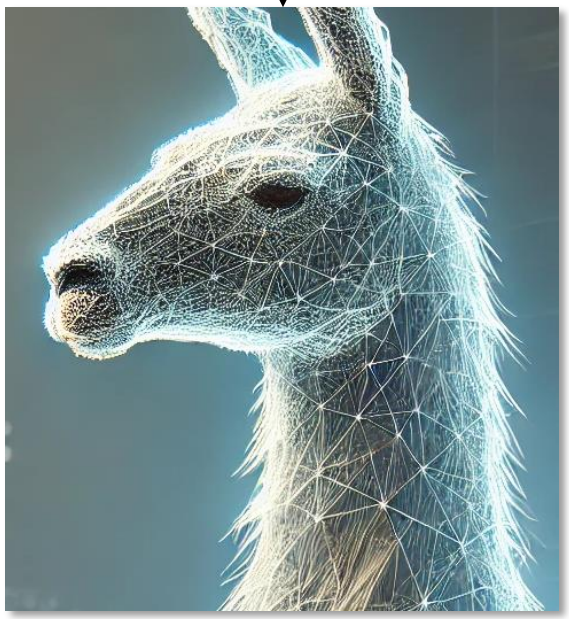
You are an expert at breaking down difficult problems into simple problems by analysing them. The user needs you to answer a complex question for them which is hard to answer directly. Answer the question by decomposing it and tell the user at the right time when the problem can be solved.

Constraints:

Forget all knowledge you've learned before and decompose the user's question based only on the user's answers. To make it easier for the user to answer, only ask one simple question at a time. You can only decompose the question, do not answer it directly. Only write one sentence for your answer containing a simple question.



"How many years elapsed between Lincoln's presidency and the release of the iPhone?"



1. When was Abraham Lincoln president?

System architecture

Question answering

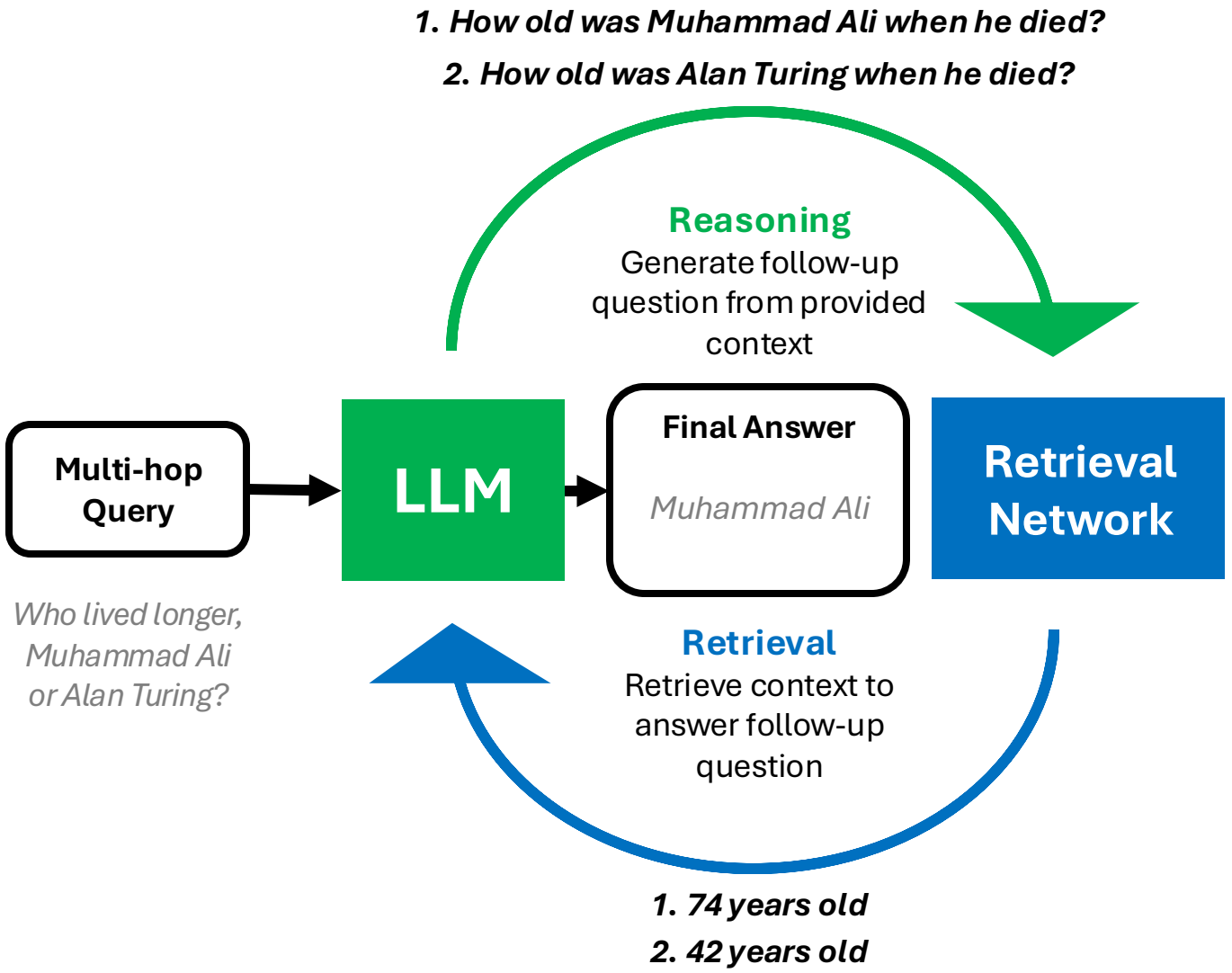
Iterative Reasoning

- How do we bring all these steps together to answer the user's original question?

1. Break the user's query into simple questions

2. Alternate between:
   - **Reasoning** (answering sub-questions from prompt)
   - **Retrieval** (retrieving answer to sub-question)

(Wu et al. 2024, Chu et al. 2024)



Answering the final question

3. Once we have all the evidences needed to answer the question, we can answer the question in its entirety using Llama again.

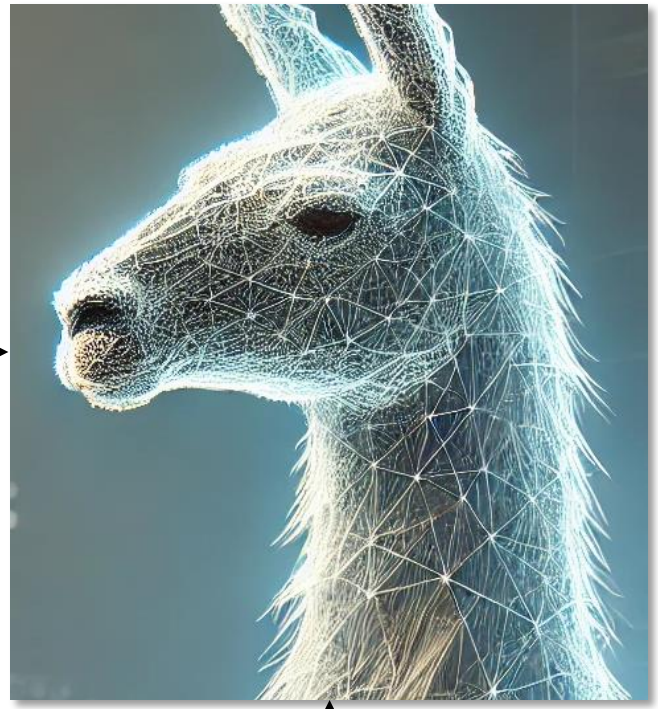
Retrieved Answers:

1. Abraham Lincoln was president from 1861 – 1865.
2. The iPhone was released in the United States on June 29, 2007.

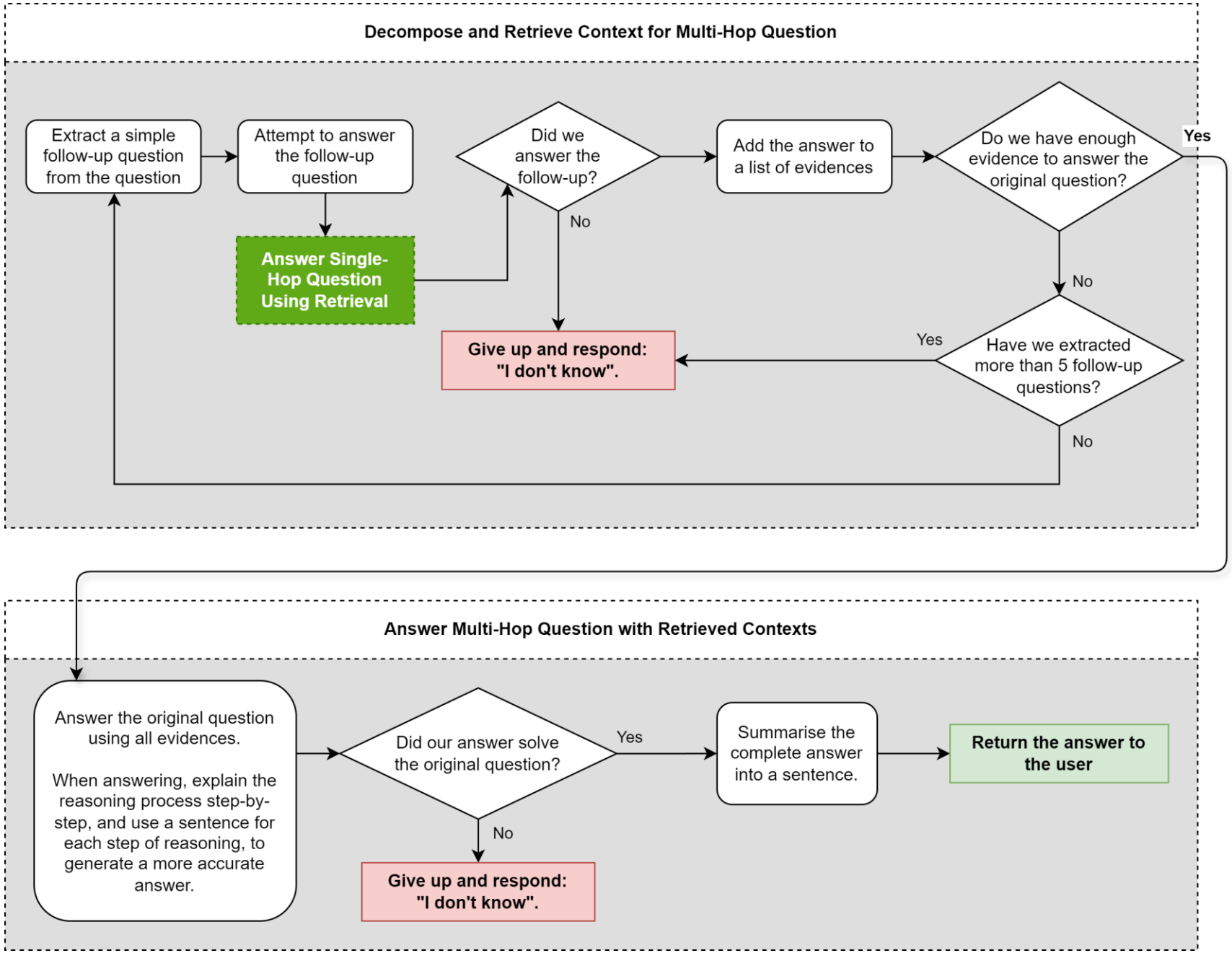
You need to answer a question given some background information. Answer the question only if the answer is present in the background information.

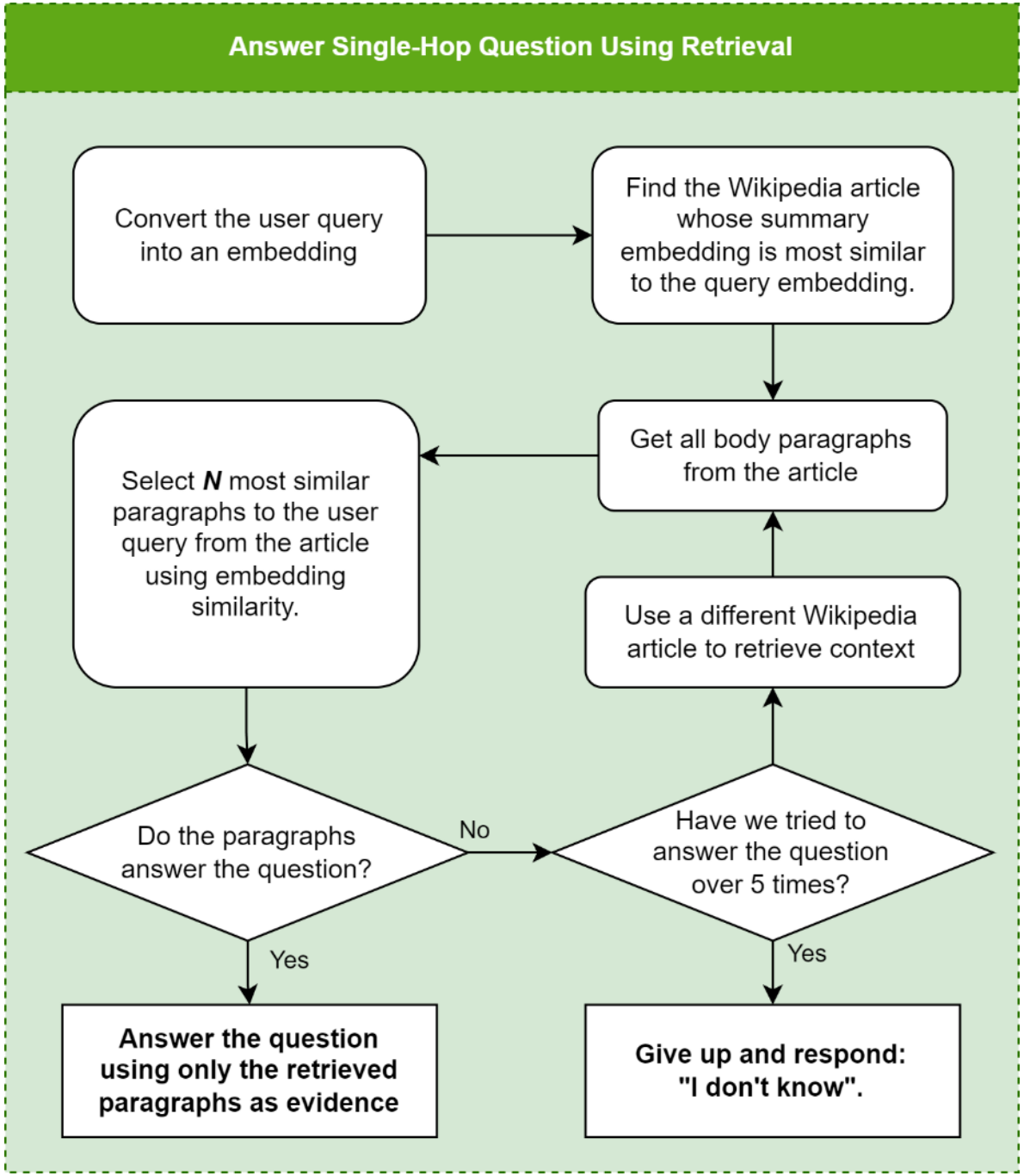
Contraints:

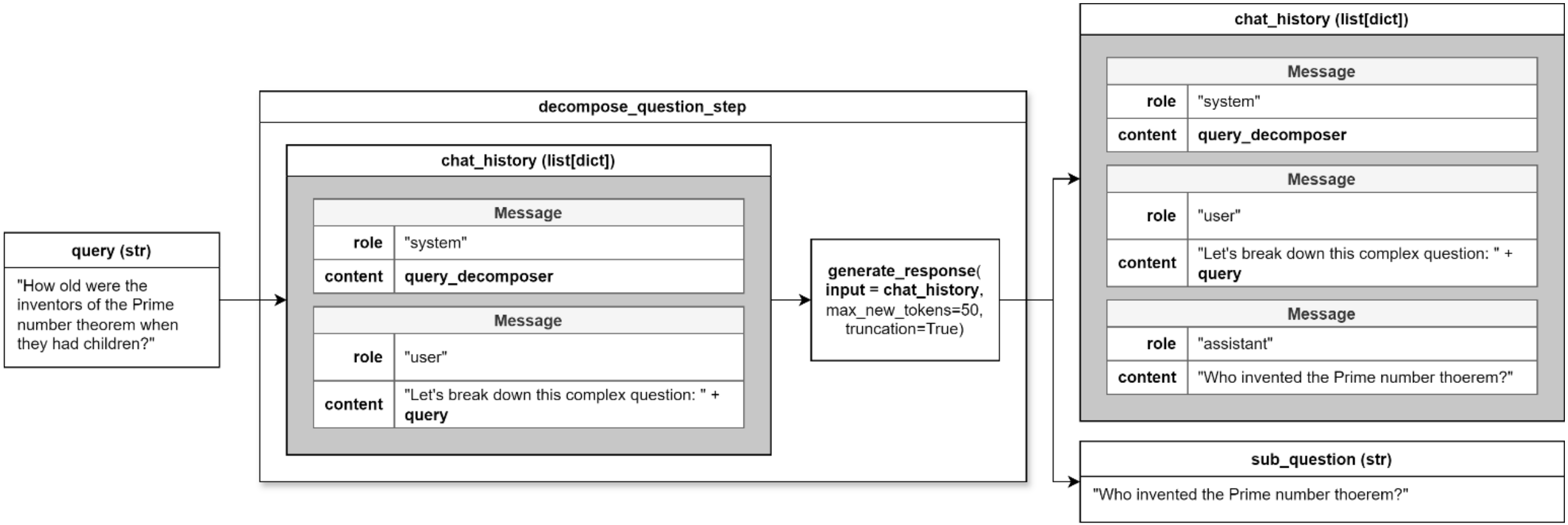
Forget all knowledge you've learned before and only answer the question using the information provided. If the background information doesn't contain the answer, say "I don't know". Explain the reasoning process for your answer in steps using sentences for each step.

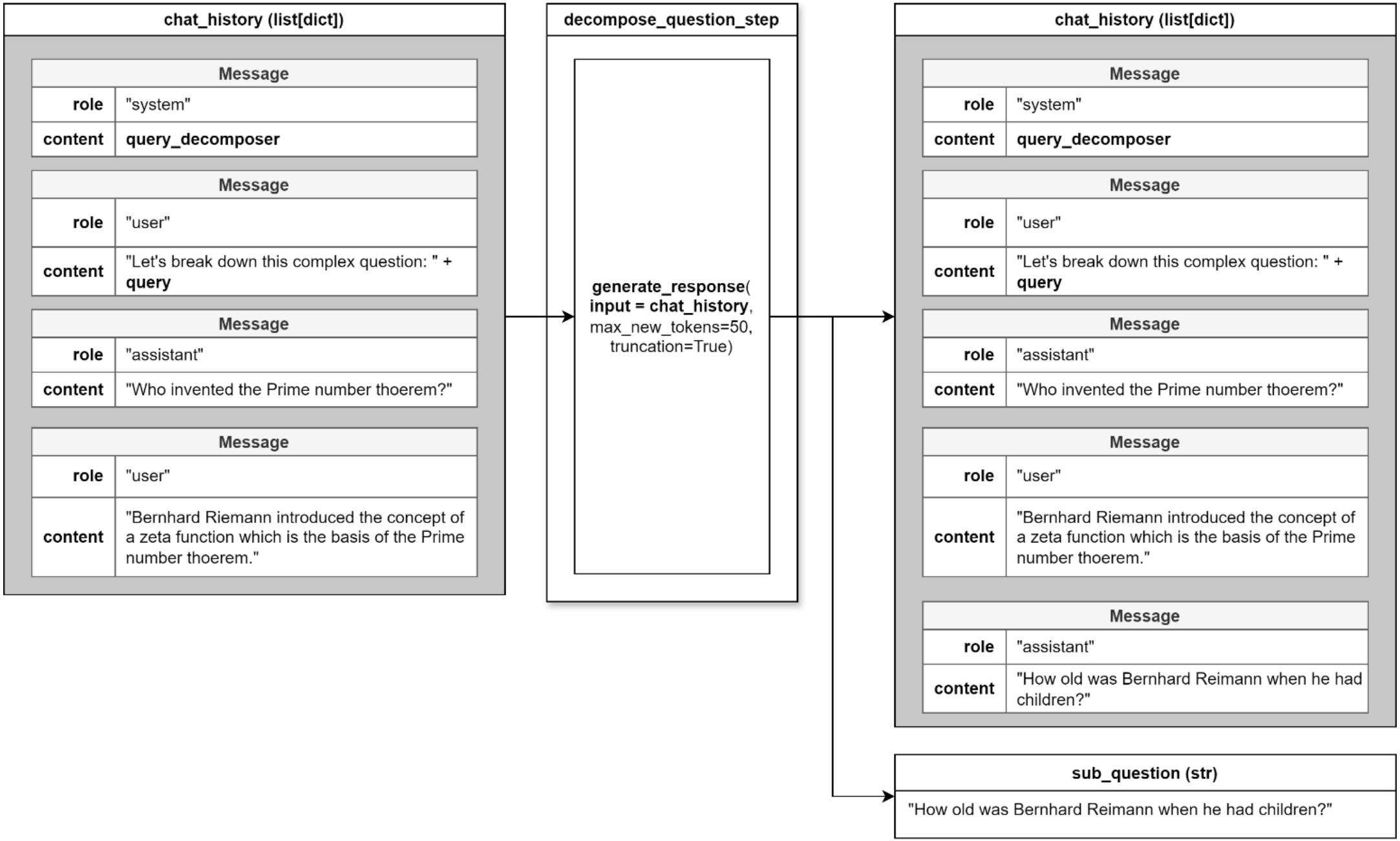


Putting it together


Decompose and Retrieve Context for Multi-Hop Question
Extract a simple follow-up question from the question
Attempt to answer the follow-up question
Answer Single-Hop Question Using Retrieval
Did we answer the follow-up?
No
Give up and respond: "I don't know".
Add the answer to a list of evidences
Do we have enough evidence to answer the original question?
Yes
No
Have we extracted more than 5 follow-up questions?
Yes
No
Answer Multi-Hop Question with Retrieved Contexts
Answer the original question using all evidences.
When answering, explain the reasoning process step-by-step, and use a sentence for each step of reasoning, to generate a more accurate answer.
Did our answer solve the original question?
Yes
No
Give up and respond: "I don't know".
Summarise the complete answer into a sentence.
Return the answer to the user


Answer Single-Hop Question Using Retrieval
Convert the user query into an embedding
Find the Wikipedia article whose summary embedding is most similar to the query embedding.
Get all body paragraphs from the article
Select N most similar paragraphs to the user query from the article using embedding similarity.
Use a different Wikipedia article to retrieve context
Do the paragraphs answer the question?
No
Have we tried to answer the question over 5 times?
Yes
Yes
Answer the question using only the retrieved paragraphs as evidence
Give up and respond: "I don't know".


query (str)
"How old were the inventors of the Prime number theorem when they had children?"
decompose_question_step
chat_history (list[dict])
Message
role
"system"
content
query_decomposer
Message
role
"user"
content
"Let's break down this complex question: " + query
generate_response(input = chat_history, max_new_tokens=50, truncation=True)
chat_history (list[dict])
Message
role
"system"
content
query_decomposer
Message
role
"user"
content
"Let's break down this complex question: " + query
Message
role
"assistant"
content
"Who invented the Prime number thoerem?"
sub_question (str)
"Who invented the Prime number thoerem?"


chat_history (list[dict])
Message
role
"system"
content
query_decomposer
Message
role
"user"
content
"Let's break down this complex question: " + query
Message
role
"assistant"
content
"Who invented the Prime number thoerem?"
Message
role
"user"
content
"Bernhard Riemann introduced the concept of a zeta function which is the basis of the Prime number thoerem."
decompose_question_step
generate_response(input = chat_history, max_new_tokens=50, truncation=True)
chat_history (list[dict])
Message
role
"system"
content
query_decomposer
Message
role
"user"
content
"Let's break down this complex question: " + query
Message
role
"assistant"
content
"Who invented the Prime number thoerem?"
Message
role
"user"
content
"Bernhard Riemann introduced the concept of a zeta function which is the basis of the Prime number thoerem."
Message
role
"assistant"
content
"How old was Bernhard Reimann when he had children?"
sub_question (str)
"How old was Bernhard Reimann when he had children?"

Conclusion

Conclusion

- We researched how hallucinations can be mitigated for LLMs.
- We have implemented a RAG system which can uses **retrieval** over a Wikipedia knowledge base to answer user queries.
- We have implemented the following components:
  - Knowledge Base
  - Embeddings
  - Retrieval
  - Query decomposition
  - Question answering

Thank You

References

Conneau, A & Kiela, D 2018, *SentEval: an Evaluation Toolkit for Universal Sentence Representations*, arXiv.
Marcus, G & Future of Life Institute 2023, *Pause Giant AI Experiments: an Open Letter*, Future of Life Institute.
LangChain 2024, *Mental Health Therapy as an LLM State Machine*, LangChain Blog.
Gates, B 2023, *AI Is about to Completely Change How You Use Computers*, gatesnotes.com.
Tonmoy, SMTI, Zaman, SMM, Jain, V, Rani, A, Rawte, V, Chadha, A & Das, A 2024, *A Comprehensive Survey of Hallucination Mitigation Techniques in Large Language Models*, arXiv.
Babu P R, D 2023, *Exploring the LLM Landscape: from Parametric Memory to Agent-Oriented Models*, Medium.
Adams, K 2024, *How Often Do LLMs Hallucinate When Producing Medical Summaries*?, MedCity News.
Hughes, S & Bae, M 2023, *Hughes Hallucination Evaluation Model (HHEM) Leaderboard*, Hugging Face, Vectara, Inc.
Li, Z, Zhang, X, Zhang, Y, Long, D, Xie, P & Zhang, M 2023, *Towards General Text Embeddings with Multi-stage Contrastive Learning*, arXiv, Cornell University.
Gao, Y, Xiong, Y, Gao, X, Jia, K, Pan, J, Bi, Y, Dai, Y, Sun, J & Wang, H 2023, *Retrieval-Augmented Generation for Large Language Models: a Survey*, arXiv.
Cao, H 2024, *Recent Advances in Text embedding: a Comprehensive Review of Top-Performing Methods on the MTEB Benchmark*, arXiv.
James Jie Pan, Wang, J & Li, G 2024, 'Survey of Vector Database Management Systems', *The VLDB Journal*, vol. 33, Springer Science+Business Media.
Chu, Z, Chen, J, Chen, Q, Wang, H, Zhu, K, Du, X, Yu, W, Liu, M & Qin, B 2024, *BeamAggR: Beam Aggregation Reasoning over Multi-source Knowledge for Multi-hop Question Answering*, arXiv.
Wu, J, Yang, L, Ji, Y, Huang, W, Karlsson, BF & Okumura, M 2024, *GenDec: a Robust Generative Question-decomposition Method for Multi-hop Reasoning*, arXiv.
Vu, T, Iyyer, M, Wang, X, Constant, N, Wei, J, Wei, J, Tar, C, Sung, Y-H, Zhou, D, Le, Q & Luong, T 2023, *FreshLLMs: Refreshing Large Language Models with Search Engine Augmentation*, arXiv.

Q & A