Lecture 8: LLM-2

AC215

Pavlos Protopapas
SEAS/Harvard

Outline

- Advanced RAG
- Agents

Tutorial 8: RAG

In this tutorial, we're building a Retrieval-Augmented Generation (RAG) system, powered by a ChromaDB vector database and a Large Language Model (Gemini).

For the Formaggio.me chatbot to truly earn its title as a cheese connoisseur, it needs to go beyond the basics, knowing rare and lesser-known cheeses, along with all the juicy details. Standard LLMs won't have this specialized knowledge, so we've gathered a collection of books to build the RAG system.

And of course, the whole setup is containerized!

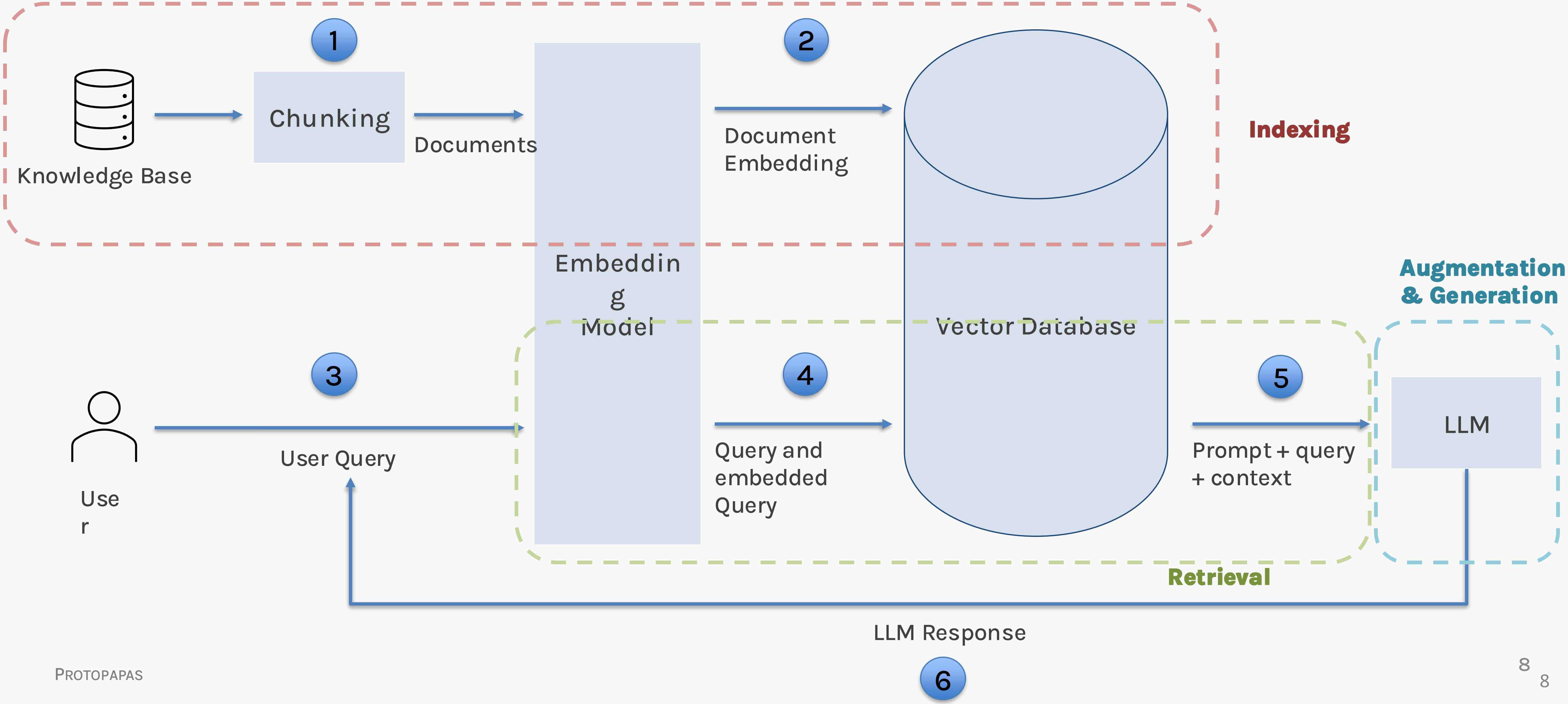
https://github.com/dlops-io/llm-rag

Outline

- Advanced RAG
    - Naïve RAG - Recap
    - Pre-Retrieval Optimization
    - Retriever Optimization
    - Post-Retrieval Optimization
    - Self-RAG
    - Corrective-RAG
- Agents

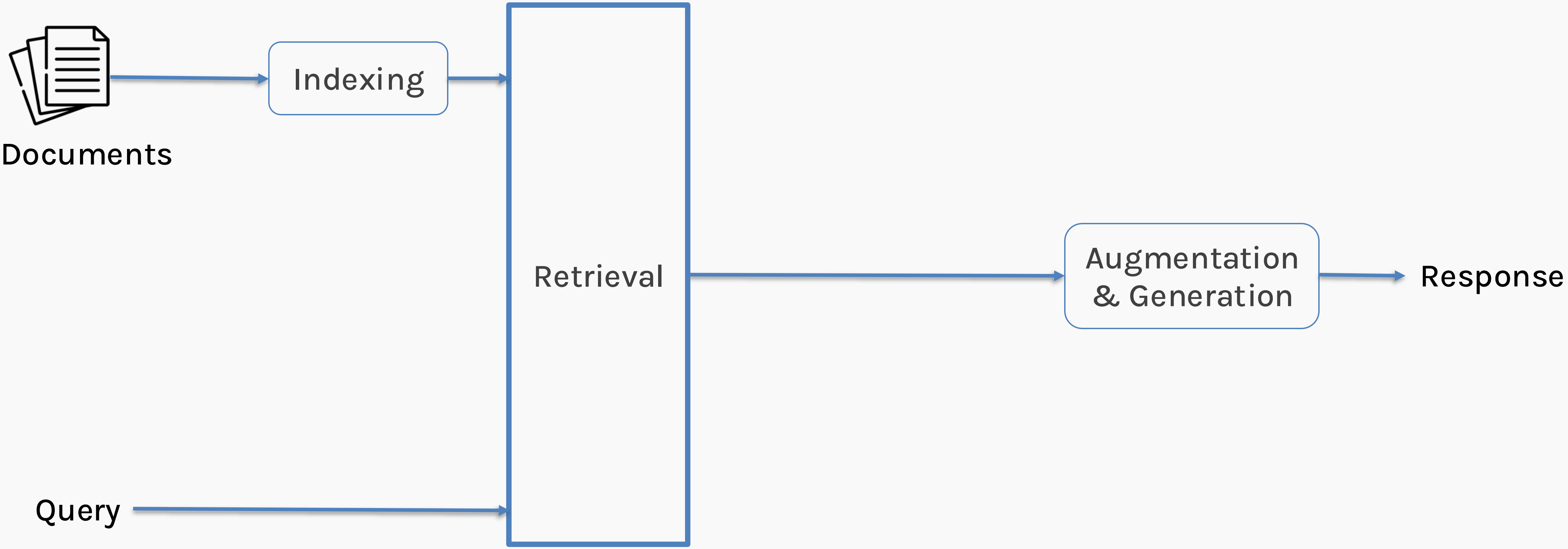
Naïve RAG - Recap

Naïve RAG - Recap



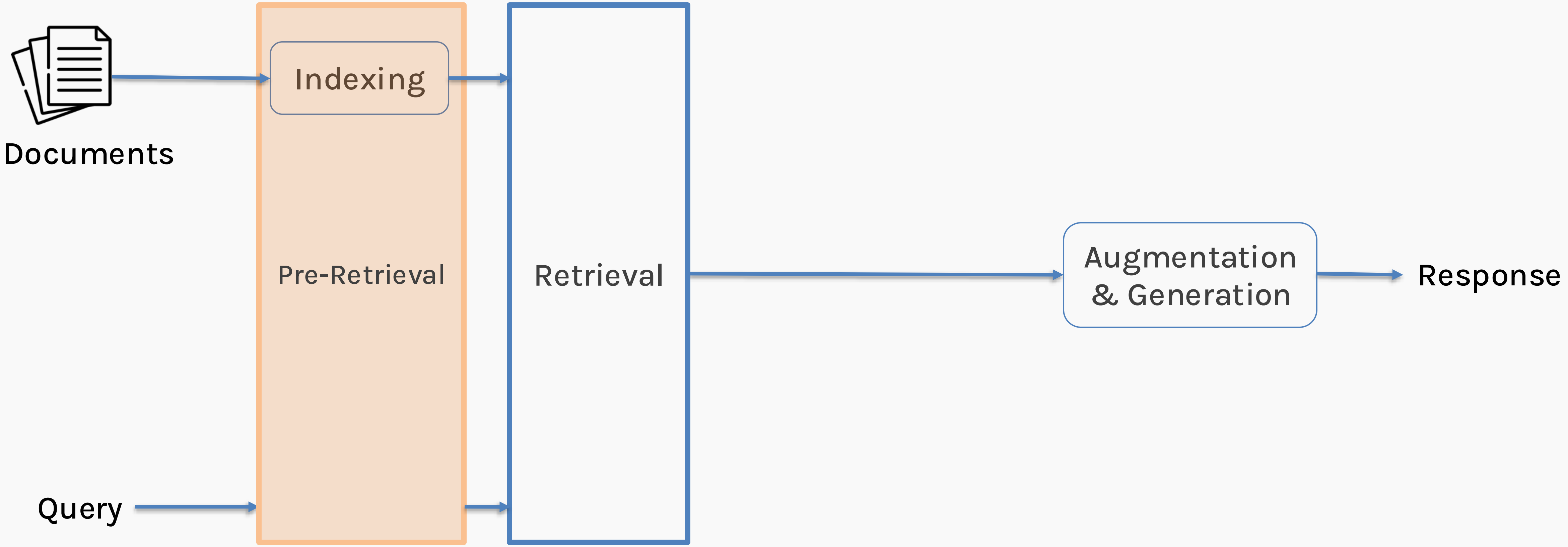
Naïve RAG

Now, let's look at the big picture.



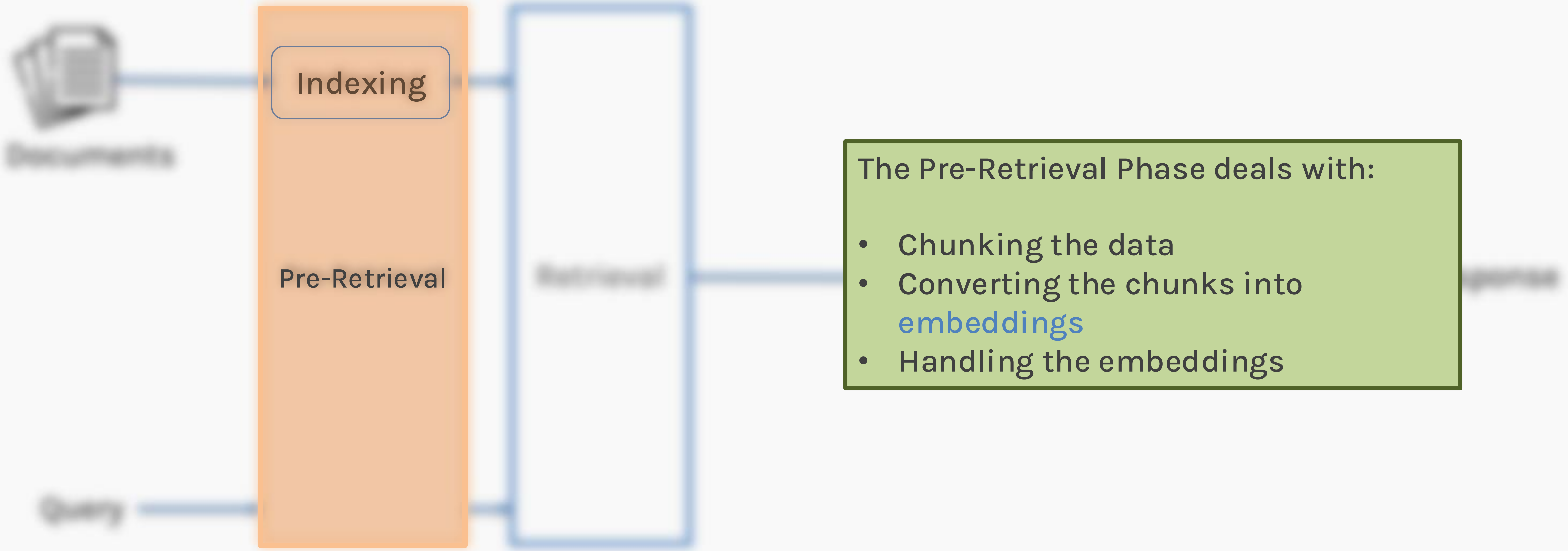
Naïve RAG

Now, let's look at the big picture.



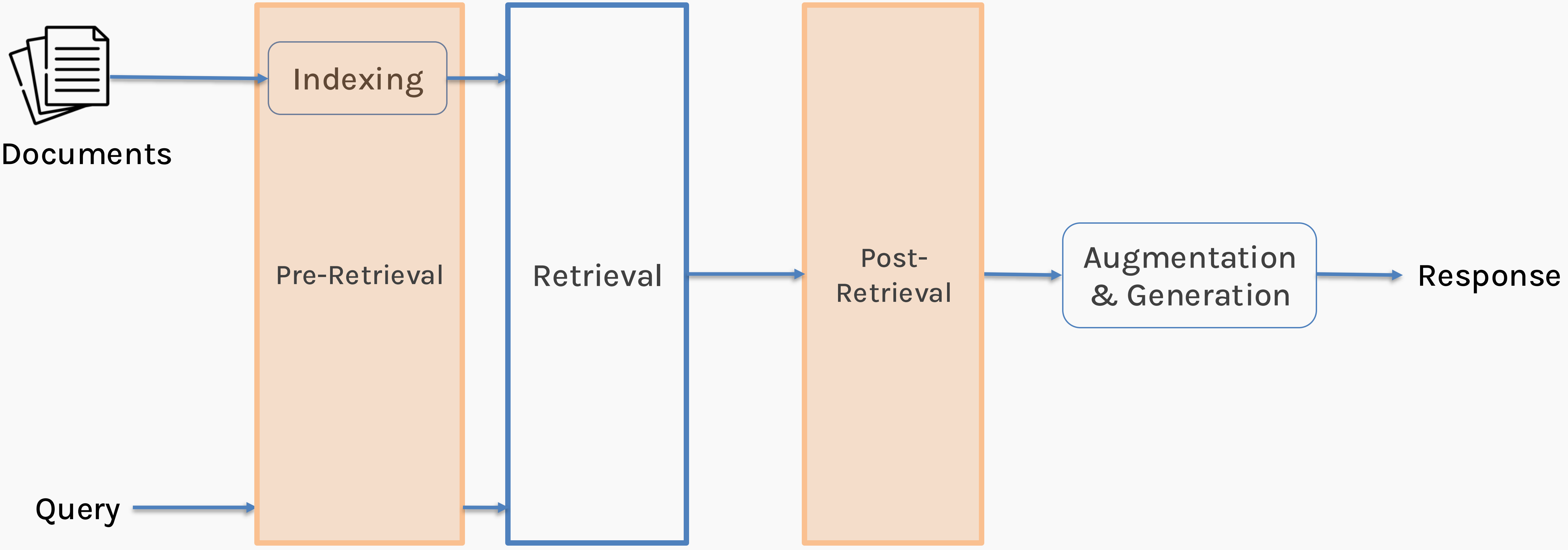
Naive RAG

Now, let's look at the big picture.



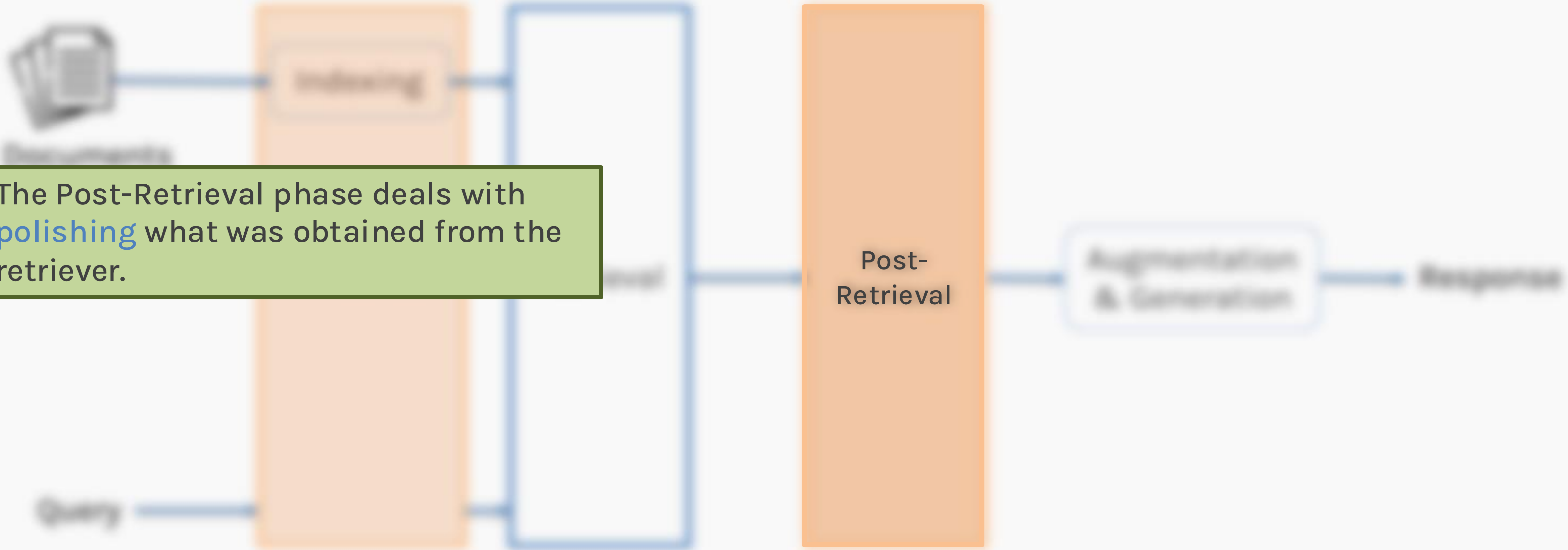
Naïve RAG

Now, let's look at the big picture.



Naive RAG

Now, let's look at the big picture.



The Post-Retrieval phase deals with polishing what was obtained from the retriever.

Outline

- Naïve RAG - Recap
- **Pre-Retrieval Optimization**
- Retriever Optimization
- Post-Retrieval Optimization
- Self-RAG
- Corrective-RAG

Pre-Retrieval Optimization

Pre-Retrieval Optimization

The pre-retrieval stage can be optimized in many ways.

We will be looking at 2 ways of doing so:

1. Indexing
2. Query Manipulation

Pre-Retrieval Optimization - Indexing

I. Improve the chunking process

By default, we do character splitting for the chunks. For example, if we have a document that says:

> Machine learning is a subset of artificial intelligence that focuses on building systems that learn from data. It is used in various applications such as recommendation engines, autonomous vehicles, and predictive analytics.

Pre-Retrieval Optimization - Indexing

I. Improve the chunking process

Machine learning is a subset of artificial intelligence that focuses on building systems that learn from data. It is used in various applications such as recommendation engines, autonomous vehicles, and predictive analytics.

By using character splitting of chunk size 50, the chunks would be:

a. "Machine learning is a subset of artificial intelli”
b. "gence that focuses on building systems that lear”
c. "n from data. It is used in various applications ”
d. "such as recommendation engines, autonomous vehic"
e. "les, and predictive analytics."

Do you think this is a **good** chunk?

Pre-Retrieval Optimization - Indexing

Let's take another example

Advancements in Transfer Learning for NLP

Abstract:
"Transfer learning has become a crucial technique in NLP. This paper explores recent advancements, including fine-tuning pre-trained models like BERT and GPT-3, and domain adaptation methods. Our experiments demonstrate significant improvements in performance across various NLP tasks."

Methodology:
"We fine-tuned BERT and GPT-3 models on specific NLP tasks, adapting them to different domains. Domain adaptation involved additional pre-training on domain-specific data. Our approach leverages the pre-trained knowledge and adapts it to new tasks, achieving higher accuracy and efficiency."

Results:
"The results indicate a 20% increase in accuracy for domain-specific tasks using our fine-tuning and domain adaptation techniques. We observed substantial performance gains compared to baseline models."

Model research paper

Now, if we do character splitting for chunks (chunk size=200), we get:

Chunk 1:
Recent techniques in transfer learning for NLP Abstract: Transfer learning has become a crucial technique in NLP. This paper explores recent advancements, including fine-tuning pre-trained models like BERT and GPT-3, and dom

Chunk 2:
ain adaptation methods. Our experiments demonstrate significant improvements in performance across various NLP tasks. Methodology: We fine-tuned BERT and GPT-3 models on specific NLP tasks, adapting them to different do

....

Pre-Retrieval Optimization - Indexing

Let’s take another example

nks (chunk

Advanceme
Learning for

Abstract:
"Transfer learning has b
NLP. This paper explore
including fine-tuning pr
and GPT-3, and domain
experiments demonstra
in performance across

Methodology:
"We fine-tuned BERT ar
NLP tasks, adapting the
Domain adaptation inv
on domain-specific dat
the pre-trained knowle
tasks, achieving higher

Results:
"The results indicate a
domain-specific tasks
domain adaptation tec
substantial performanc
baseline models."

stract: Transfer
his paper
ing pre-trained

strate
various NLP
3 models on

Model research paper

Pre-Retrieval Optimization - Indexing

Let's take another example

Advancements in Transfer Learning for NLP

Abstract:
"Transfer learning has become a crucial technique in NLP. This paper explores recent advancements, including fine-tuning pre-trained models like BERT and GPT-3, and domain adaptation methods. Our experiments demonstrate significant improvements in performance across various NLP tasks."

Methodology:
"We fine-tuned BERT and GPT-3 models on specific NLP tasks, adapting them to different domains. Domain adaptation involved additional pre-training on domain-specific data. Our approach leverages the pre-trained knowledge and adapts it to new tasks, achieving higher accuracy and efficiency."

Results:
"The results indicate a 20% increase in accuracy for domain-specific tasks using our fine-tuning and domain adaptation techniques. We observed substantial performance gains compared to baseline models."

Model research paper

Now, if we do character splitting for chunks (chunk size=200), we get:

Chunk 1:
Recent techniques in transfer learning for NLP Abstract: Transfer learning has become a crucial technique in NLP. This paper explores recent advancements, including fine-tuning pre-trained models like BERT and GPT-3, and **dom**

Chunk 2:
ain adaptation methods. Our experiments demonstrate significant improvements in performance across various **NLP tasks. Methodology:** We fine-tuned BERT and GPT-3 models on specific NLP tasks, adapting them to different do

....

Pre-Retrieval Optimization - Indexing

Let's take another example

Advancements in Transfer Learning for NLP

Abstract:
"Transfer learning has become a crucial technique in NLP. This paper explores recent advancements, including fine-tuning pre-trained models like BERT and GPT-3, and domain adaptation methods. Our experiments demonstrate significant improvements in performance across various NLP tasks."

Methodology:
"We fine-tuned BERT and GPT-3 models on specific NLP tasks, adapting them to different domains. Domain adaptation involved additional pre-training on domain-specific data. Our approach leverages the pre-trained knowledge and adapts it to new tasks, achieving higher accuracy and efficiency."

Results:
"The results indicate a 20% increase in accuracy for domain-specific tasks using our fine-tuning and domain adaptation techniques. We observed substantial performance gains compared to baseline models."

Model research paper

The optimal way to do it would be:

Chunk 1:
Advancements in Transfer Learning for NLP

Chunk 2:
Abstract:
Transfer learning has become a crucial technique in NLP. This paper explores recent advancements, including fine-tuning pre-trained models like BERT and GPT-3, and domain adaptation methods. Our experiments demonstrate significant improvements in performance across various NLP tasks.

Chunk 3:
Methodology:
We fine-tuned BERT and GPT-3 models on specific NLP tasks, adapting them to different domains. Domain adaptation involved additional pre-training on domain-specific data. Our approach leverages the pre-trained knowledge and adapts it to new tasks, achieving higher accuracy and efficiency.

....

This is called **semantic chunking**

Pre-Retrieval Optimization - Indexing

Let's take another example

Advancements in Transfer Learning for NLP

Abstract:
"Transfer learning has become a crucial technique in NLP. This paper explores recent advancements, including fine-tuning pre-trained models like BERT and GPT-3, and domain adaptation methods. Our experiments demonstrate significant improvements in performance across various NLP tasks."

Methodology:
"We fine-tuned BERT and GPT-3 models on specific NLP tasks, adapting them to different domains. Domain adaptation involved additional pre-training on domain-specific data. Our approach leverages the pre-trained knowledge and adapts it to new tasks, achieving higher accuracy and efficiency."

Results:
"The results indicate a 20% increase in accuracy for domain-specific tasks using our fine-tuning and domain adaptation techniques. We observed substantial performance gains compared to baseline models."

Model research paper

would be:

rning for NLP

a crucial technique in NLP. This paper
s, including fine-tuning pre-trained
nd domain adaptation methods. Our
nificant improvements in performance

models on specific NLP tasks, adapting
main adaptation involved additional pre-
ata. Our approach leverages the pre-
it to new tasks, achieving higher

This is called **semantic chunking**

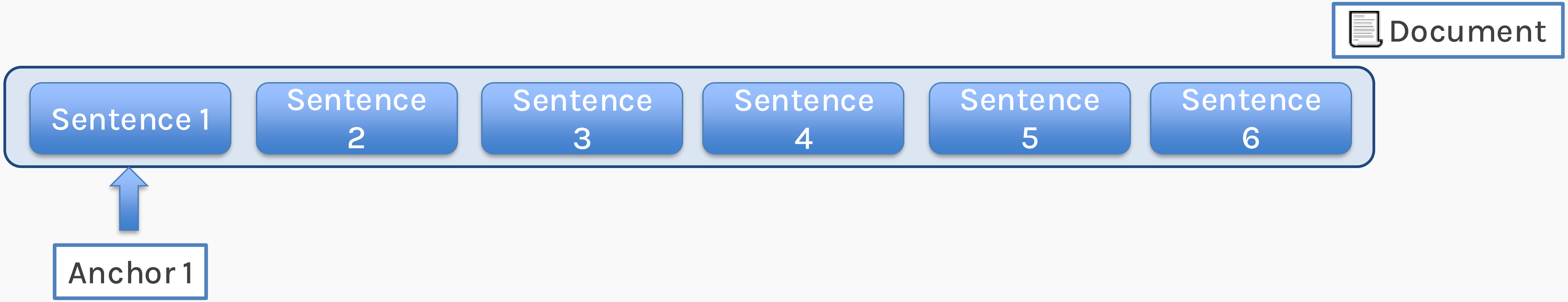
Pre-Retrieval Optimization - Indexing

Semantic Chunking – Steps

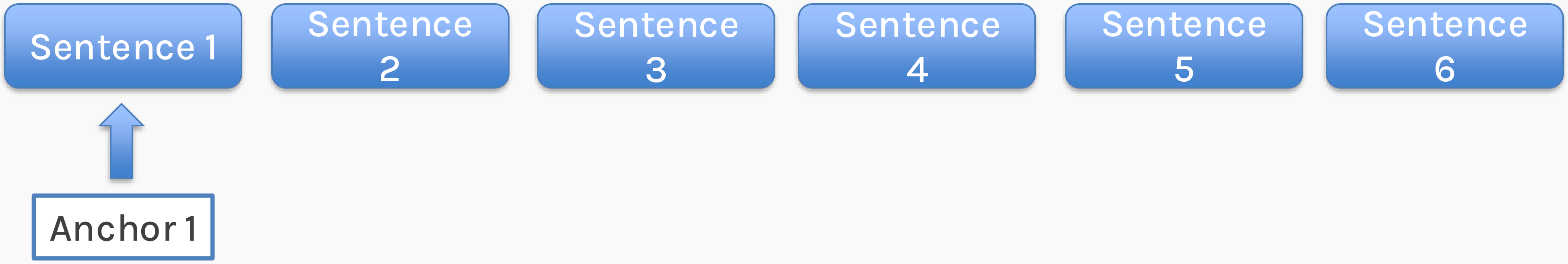
1. Splitting: We split the document to sentences using separators(.,?,!).
2. Grouping: Select anchor sentences and choose how many sentences to consider at either side of the anchor (window size).
3. Similarity Check: Calculate the distance between the group of sentences (e.g.: cosine similarity).
4. Chunking: Chunk together the similar sentences.

Confused?
Let's look at an example!

Pre-Retrieval Optimization - Indexing

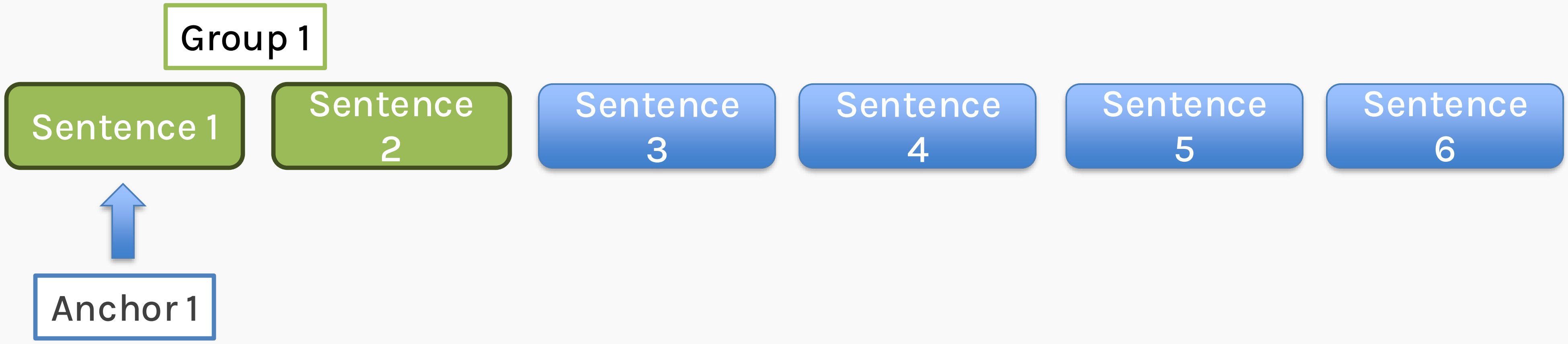


Pre-Retrieval Optimization - Indexing



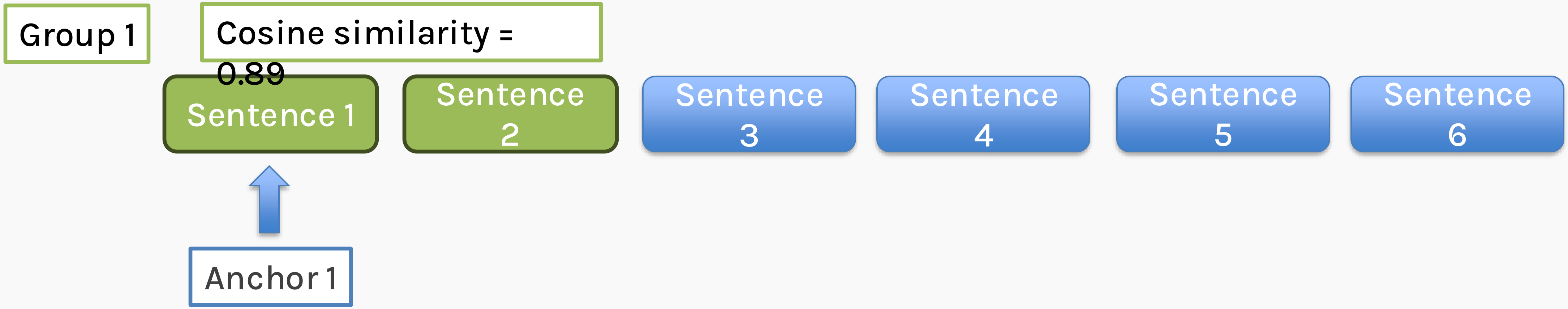
1. Let's suppose we choose the number of sentences at either side to be **1.**

Pre-Retrieval Optimization - Indexing



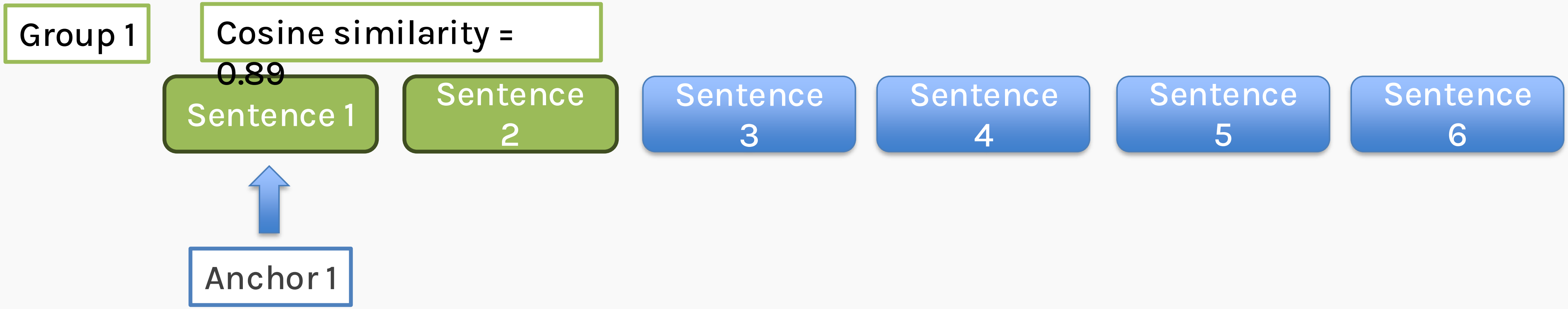
1. Let's suppose we choose the number of sentences at either side to be **1.**
2. We calculate the cosine similarity of the sentences in the group.

Pre-Retrieval Optimization - Indexing



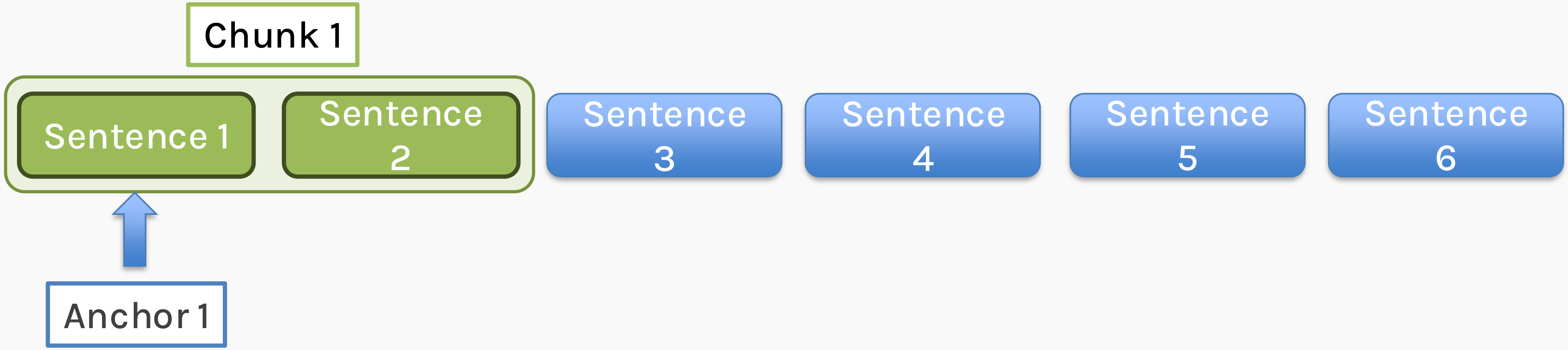
1. Let’s suppose we choose the number of sentences at either side to be **1.**
2. We calculate the cosine similarity of the sentences in the group.

Pre-Retrieval Optimization - Indexing



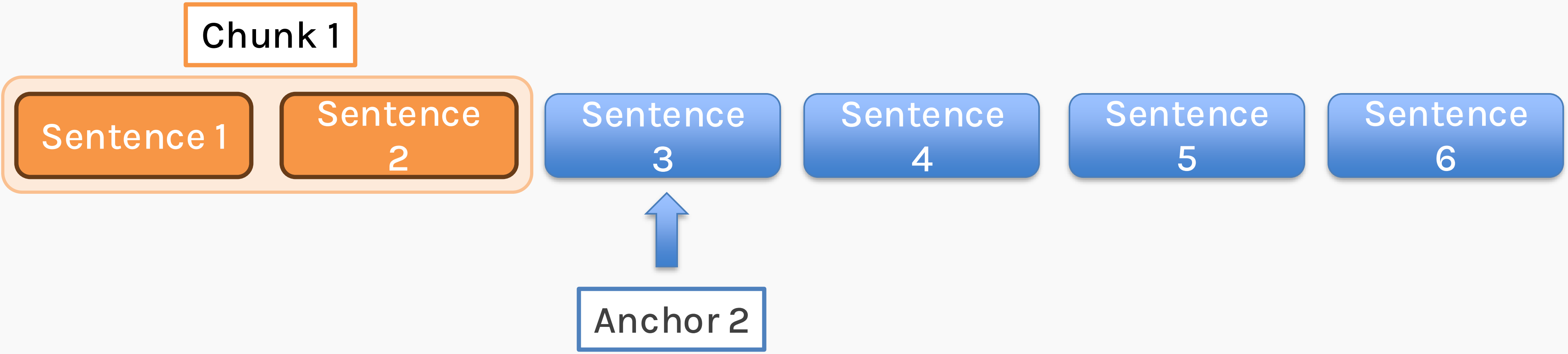
1. Let's suppose we choose the number of sentences at either side to be **1.**
2. We calculate the cosine similarity of the sentences in the group.
3. Since the cosine similarity here is high (Assuming our threshold is 0.8), we chunk together the sentences.

Pre-Retrieval Optimization - Indexing



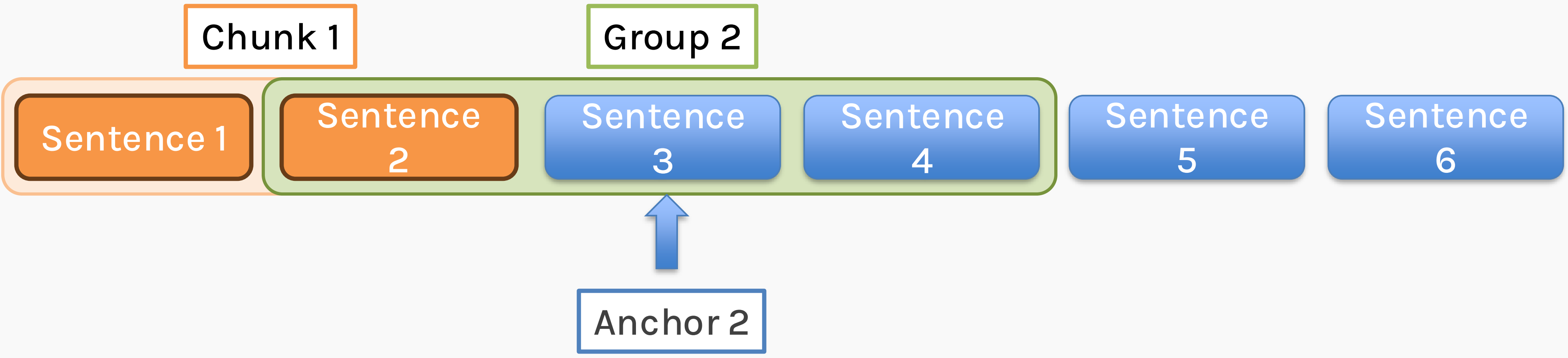
1. Let's suppose we choose the number of sentences at either side to be **1.**
2. We calculate the cosine similarity of the sentences in the group.
3. Since the cosine similarity here is high (Assuming our threshold is 0.8), we chunk together the sentences.

Pre-Retrieval Optimization - Indexing



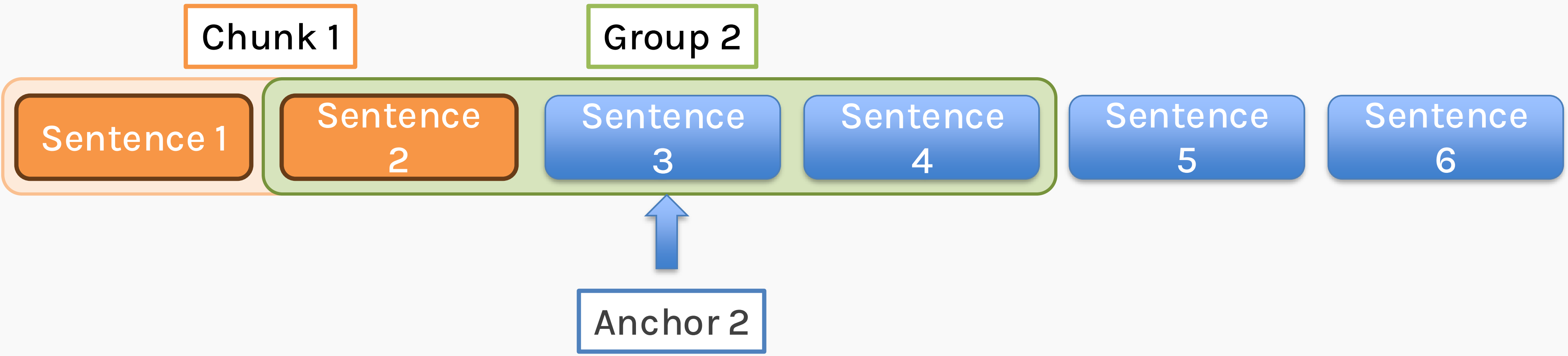
1. We then move to Anchor 2.

Pre-Retrieval Optimization - Indexing



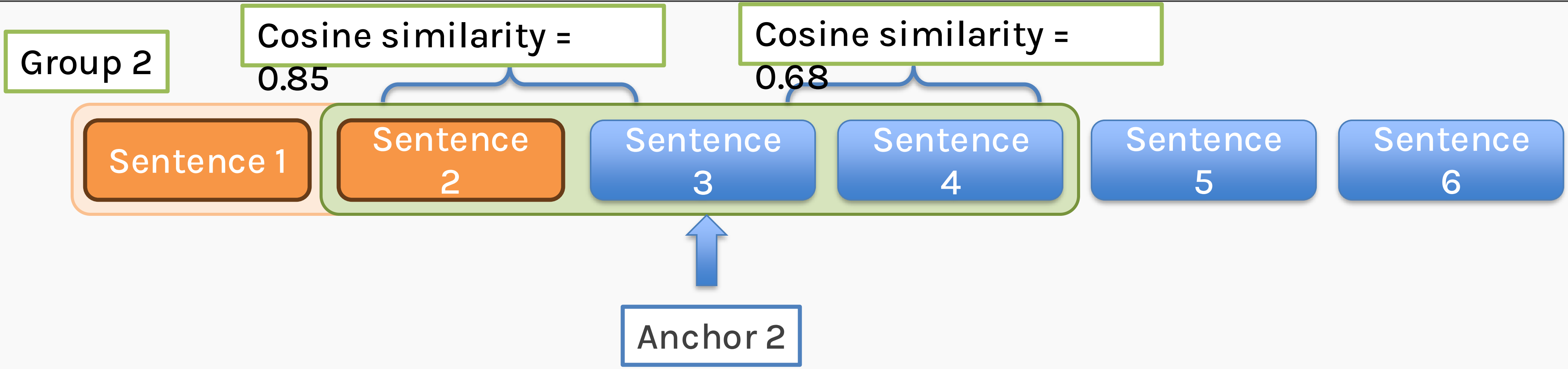
1. We then move to Anchor 2.
2. Since window size=1, we group one sentence from the left and one from the right of the anchor.

Pre-Retrieval Optimization - Indexing



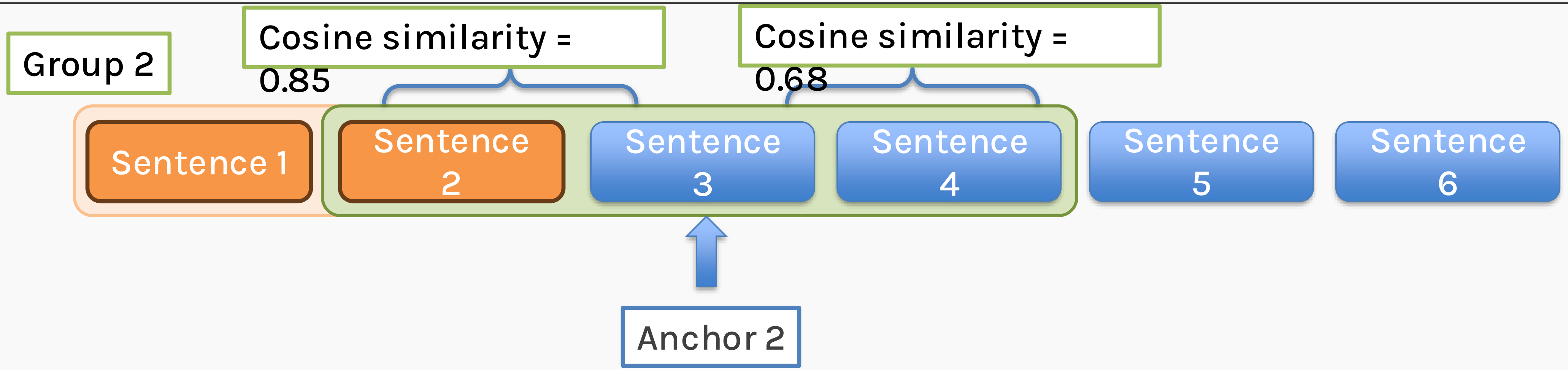
1. We then move to Anchor 2.
2. Since window size=1, we group one sentence from the left and one from the right of the anchor.
3. We then calculate the similarity indices of the sentences in Group 2 with Anchor 2.
   (Threshold=0.8)

Pre-Retrieval Optimization - Indexing



1. We then move to Anchor 2.
2. Since window size=1, we group one sentence from the left and one from the right of the anchor.
3. We then calculate the similarity indices of the sentences in Group 2 with Anchor 2.
   (Threshold=0.8)

Pre-Retrieval Optimization - Indexing

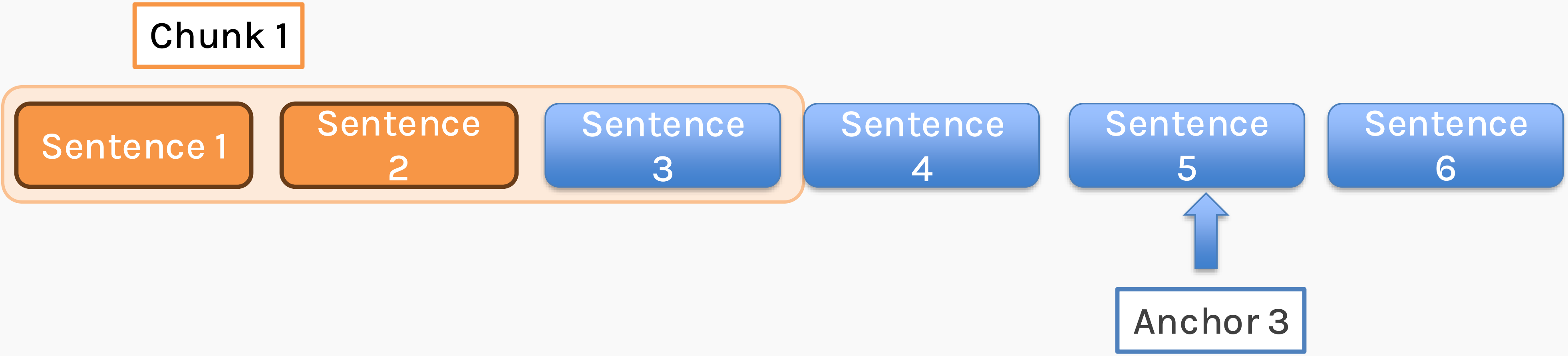


Since,

cosine similarity of Sentence 2 & Anchor 2 > 0.8 and
cosine similarity of Sentence 3 & Anchor 2 < 0.8,

we chunk together Sentence 3 into Chunk 1.

Pre-Retrieval Optimization - Indexing



Since Chunk 1 is complete.

We now move on to make the 2nd Chunk.

The anchor moves on to Sentence 5 and the process continues till we reach the end of the sentence.

Pre-Retrieval Optimization – Query Manipulation

Pre-Retrieval Optimization – Query Manipulation

2 problems can come up when it comes to queries provided by a user:

1. The query is ‘cluttered’.
   This can be due to it being sprinkled with a lot of irrelevant information.

2. The query is ambiguous.
   The query doesn’t have sufficient information.

Pre-Retrieval Optimization – Query Manipulation

1. The query is ‘cluttered’ for our RAG system

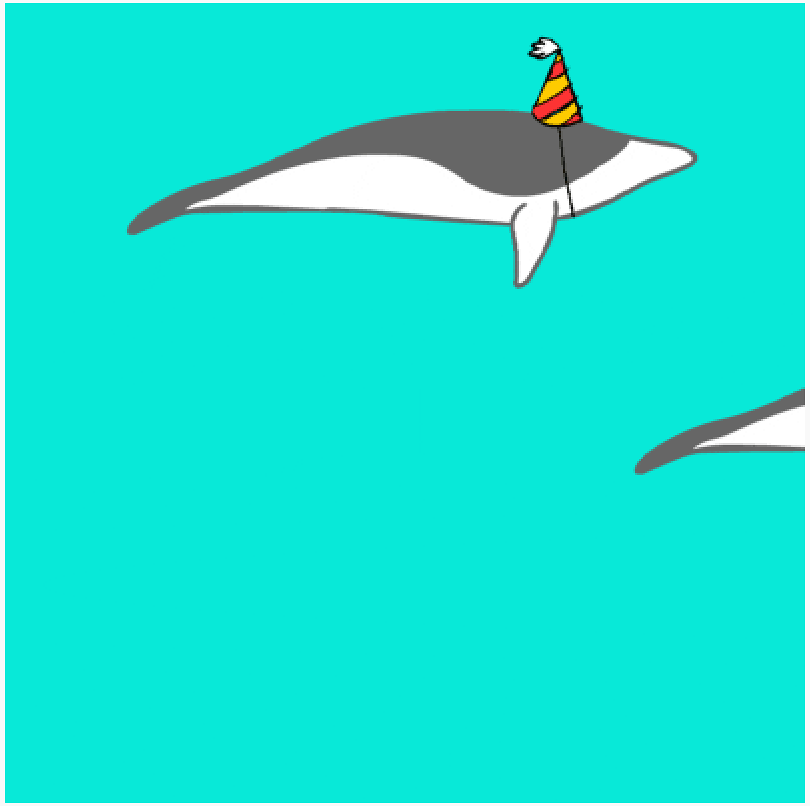
❓Original Query

We have an essay due tomorrow. We have to write about some animal. I love penguins. I could write bout them. But I could also write about dolphins. Are they animals? Maybe. Let’s do dolphins. Where do they live, for example?

🧠Rewritten query

Where do dolphins live?

The rewritten query is now **concise** and “**to the point**”.



Pre-Retrieval Optimization – Query Manipulation

2. The query is ambiguous for our RAG system.

Imagine a use case - we have created a RAG system on top of a Microsoft annual report.

❓Original Query

In what sense?
Retirements?
Promotions?

"Was there significant turnover in the executive team?"

Too broad a term.
Directors? Senior Vice Presidents?

The query is too 'narrow' and lacks information.

Pre-Retrieval Optimization – Query Manipulation

❓Original Query	"Was there significant turnover in the executive team?"
🧠 Rewritten query	Was there significant turnover in the executive team? Has there been a notable level of turnover among the executive leadership team recently? Specifically, I am interested in understanding whether multiple key positions within the executive team have experienced changes in leadership, including CEOs, CFOs, or other top executives, over the past year. Additionally, what factors contributed to these changes?

This query is completely made up by the LLM and has nothing to do with the Microsoft annual report

Outline

- Naïve RAG - Recap
- Pre-Retrieval Optimization
- **Retrieval Optimization**
- Post-Retrieval Optimization
- Self-RAG
- Corrective-RAG

Retrieval Optimization

Retrieval Optimization – Hybrid Search

Instead of just using semantic search using vectors, we can also do some keyword matching.

For e.g. If we have the sentence: Ignacio went to the bank of the river in the morning

Hybrid search in retrieval optimization combines different retrieval models to leverage the strengths of each and provide more relevant search results.

The vector search may help us disambiguate the meaning of the word ‘bank’ while the keyword matching may help us find documents related to ‘Ignacio’.

Retrieval Optimization – Hybrid Search

How do we do it?

Usually, we use two different retrievers, one for keyword search (BM25) and one for semantic matching (vector similarity).

BM25

A probabilistic retrieval model that ranks documents based on the frequency of query terms in the document.

The formula is based on TF-IDF.

To learn more about BM25: Click [here]

Retrieval Optimization – Hybrid Search

Let's say we got the following 3 paragraphs:

Original Sentence:
Ignacio went to the bank of the river in the morning

Paragraphs

Ignacio went to the riverbank early in the morning

The sentence is almost the same as the original sentence

In the morning, Ignacio went to the bank by the river to borrow some money

The bank in this sentence is completely different to the one in the original sentence!

Ignacio went to the river for swimming and splashing around. Afterwards, he lay on the riverbank, drying off in the sun.

It has words that relate to the original sentence.

Retrieval Optimization – Hybrid Search

Let's now look at the rankings given by the algorithms.

Original Sentence:
Ignacio went to the bank of the river in the morning

Paragraphs	BM25	Vector Search
Ignacio went to the riverbank early in the morning	2	1
In the morning, Ignacio went to the bank by the river to borrow some money	1	3
Ignacio went to the river for swimming and splashing around. Afterwards, he lay on the riverbank, drying off in the sun.	3	2

Retrieval Optimization – Hybrid Search

How do we combine the results?

We use one of the rank fusion techniques:

Reciprocal Rank Fusion (RRF) = $\sum_{j=1}^{n} w_j \; * \; \frac{1}{k+r(d)}$

Where,

n=number of rankings
$r(d)$ = rank of the document

These are hyper-parameters $\left\{ \begin{array}{l} w_j\text{=weight of the ranking metric} \\ k\text{=ranking constant} \end{array} \right.$

Retrieval Optimization – Hybrid Search

We take the k to be 0 and w_j=0.5

How do we combine the results?

Paragraphs	BM25	Vector Search	Reciprocal Rank Fusion (RRF)
Ignacio went to the riverbank early in the morning	2	1	$0.5{*}{}^{1}/_{2} + 0.5{*}\,{}^{1}/_{1} = 0.75$
In the morning, Ignacio went to the bank by the river to borrow some money	1	3	$0.5{*}{}^{1}/_{1} + 0.5{*}\,{}^{1}/_{3} = 0.67$
Ignacio went to the river for swimming and splashing around. Afterwards, he lay on the riverbank, drying off in the sun.	3	2	$0.5{*}{}^{1}/_{3} + 0.5{*}\,{}^{1}/_{2} = 0.42$

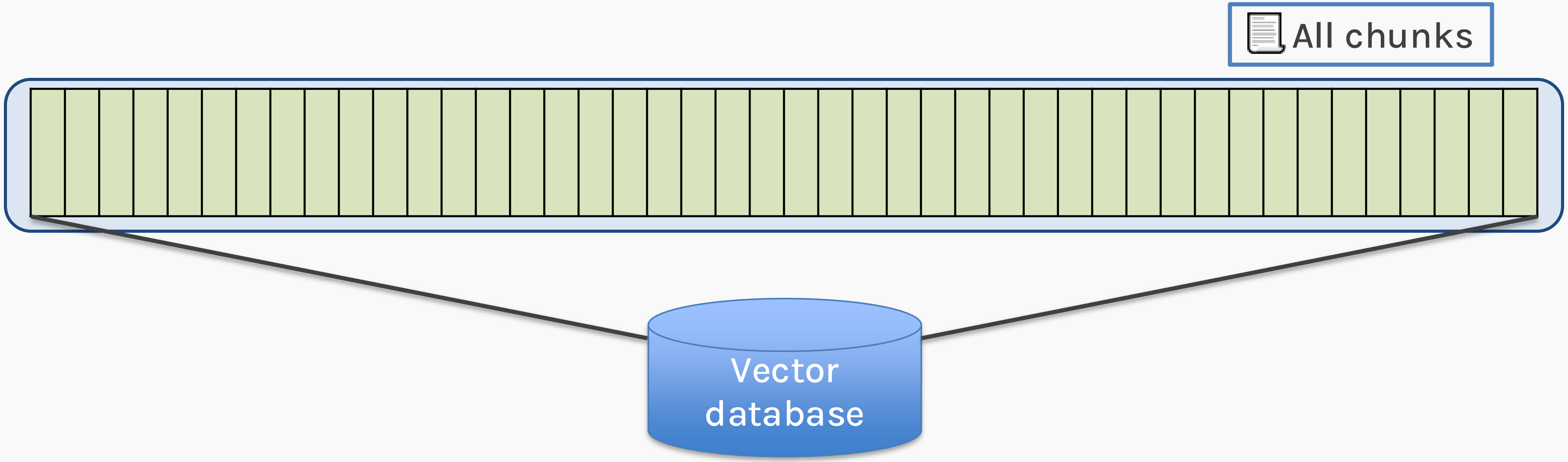
Outline

- Naïve RAG - Recap
- Pre-retrieval Optimization
- Retrieval Optimization
- **Post-Retrieval Optimization**
- Self-RAG
- Corrective-RAG

Post-Retrieval Optimization

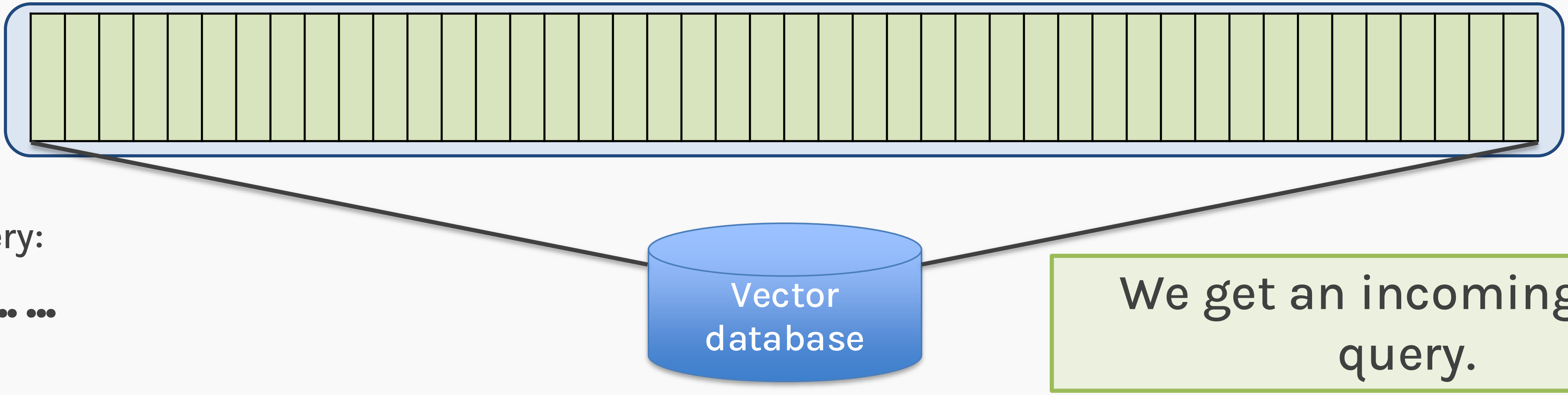
Post-Retrieval – Re-ranking

Let us go back to the bigger picture. Consider we have all the chunks stored in our vector database.



Post-Retrieval – Re-ranking

All chunks

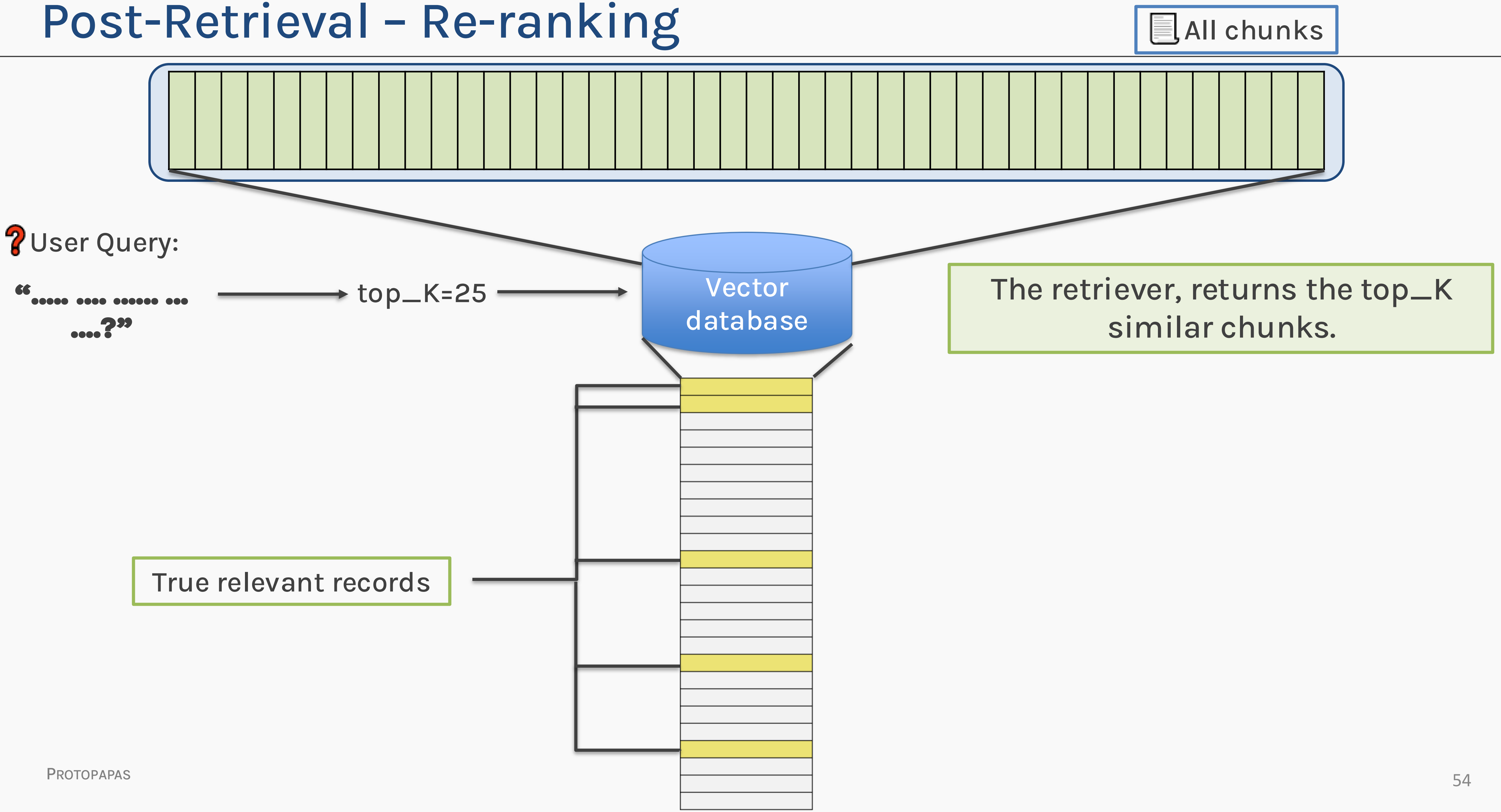


❓User Query:

"..... ....
....?"

We get an incoming user query.

Post-Retrieval – Re-ranking



Post-Retrieval – Re-ranking

All chunks

❓User Query:

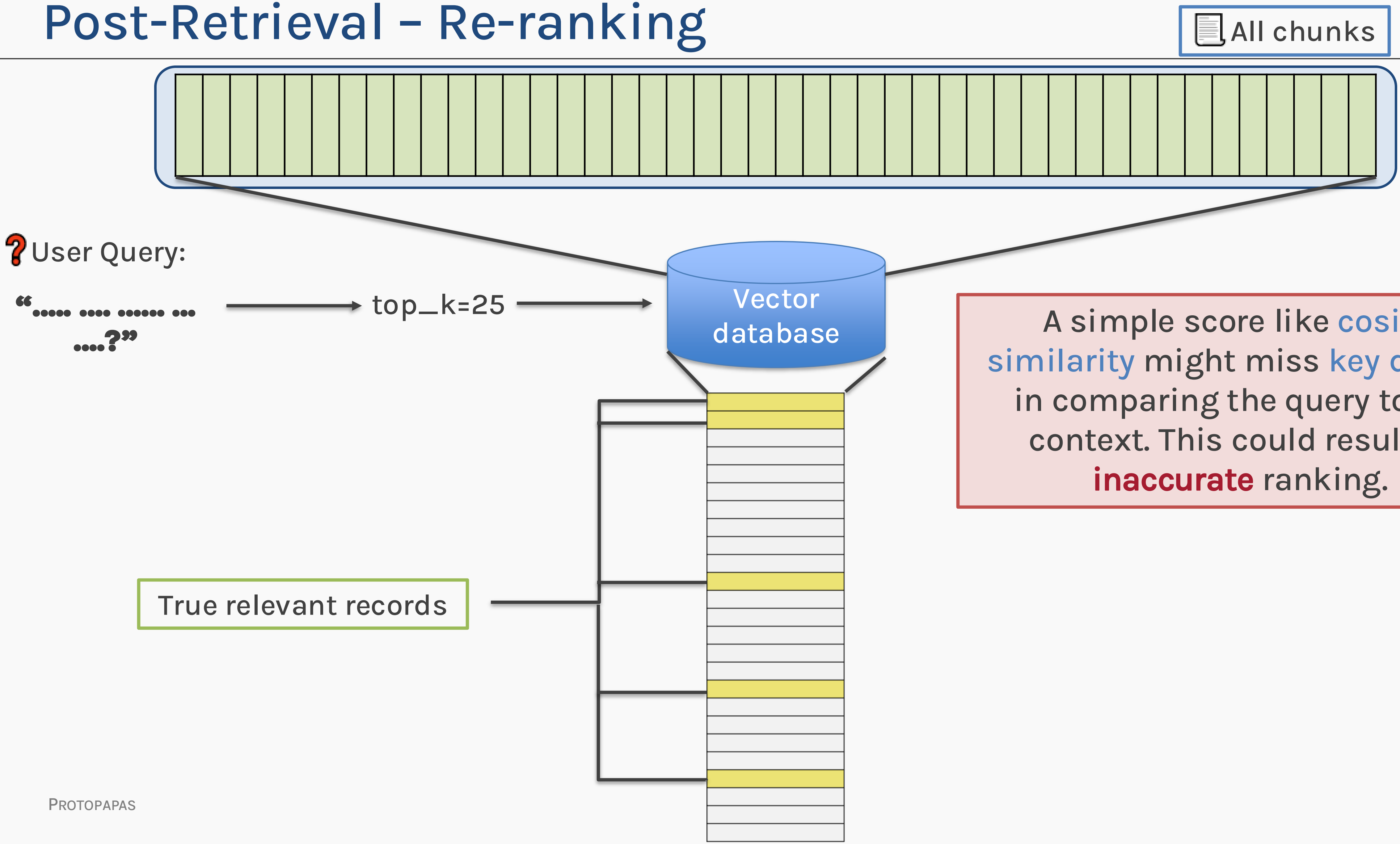
"..... ....?"

top_k

...etriever, returns the top_k similar chunks.

True relevant records

Post-Retrieval – Re-ranking



A simple score like cosine similarity might miss key details in comparing the query to the context. This could result in **inaccurate** ranking.

Post-Retrieval – Re-ranking

All chunks

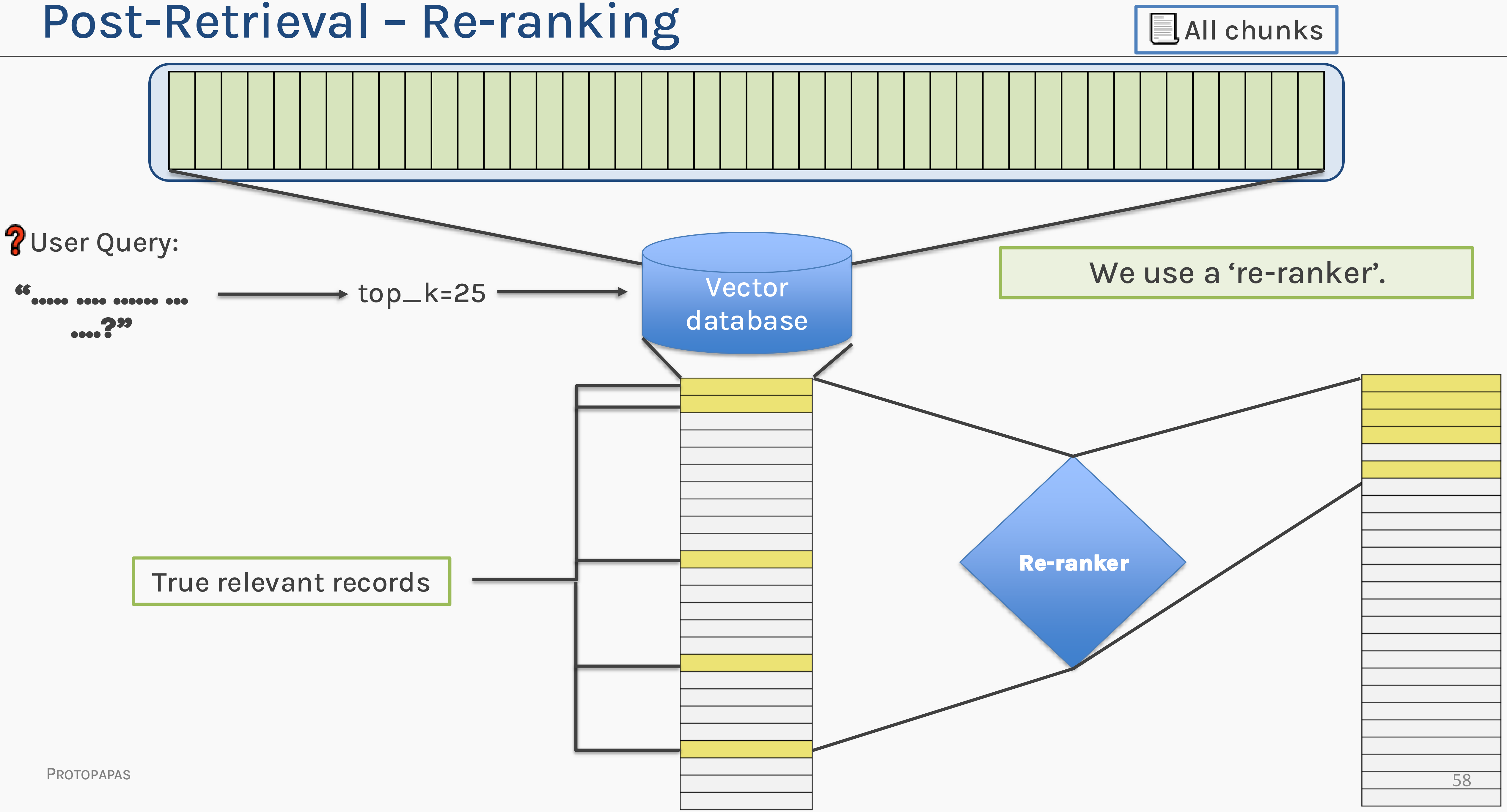
?User Query:

"..... ....
....?"

True relevant r

core like cosine
ht miss key details
g the query to the
s could result in
rate ranking.

Post-Retrieval – Re-ranking



Post-Retrieval – Re-ranking

A question that may pop up in your mind is:

Why don't we just re-rank from the very beginning instead of retrieving and then re-ranking?

The answer to that question is:

and
e need

It is also referred to as **bi-encoders** because one is used for **encoding the query** and the other for **document/chunks.**

Let us see why!

- We used encoders (retrievers) to compress all the records into vectors.
- Bi-encoders have no context on the query because we create these vectors before user query time.

Post-Retrieval – Re-ranking

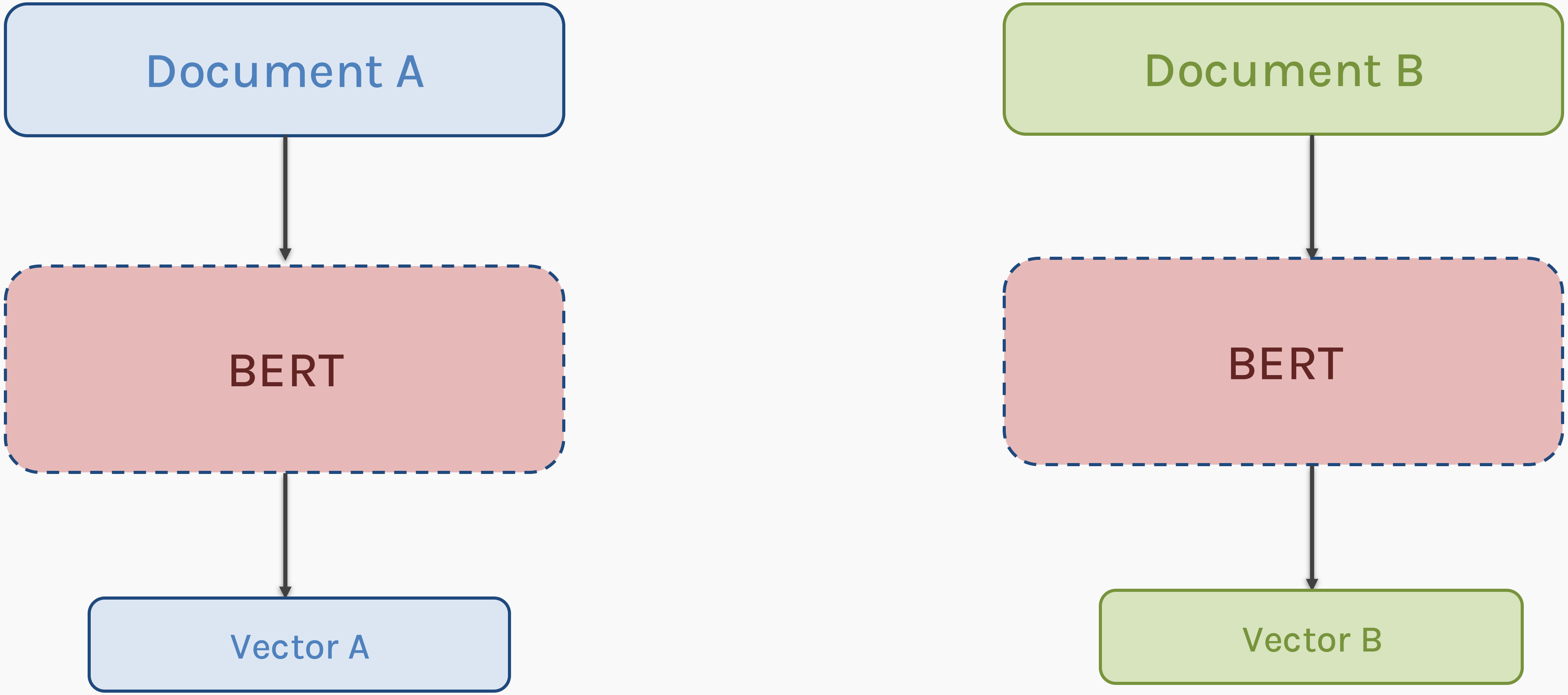


Fig: Encoder

The bi-encoder provides us with the vectors stored in the vector database.

Post-Retrieval – Re-ranking

- We thus frontload all the heavy computations when we create the initial vectors.
- Thus, when a user sends a query, we have the vectors ready. All we need to do is:
  - Run bi-encoder once to create query vector.
  - Compare query vector to document vectors with cosine similarity.

This is the main reason why retrieving is faster.

The main drawback to this is information loss which is mitigated to some extent when we use a re-ranker.

We use a cross-encoders as a re-ranker.

Post-Retrieval – Re-ranking

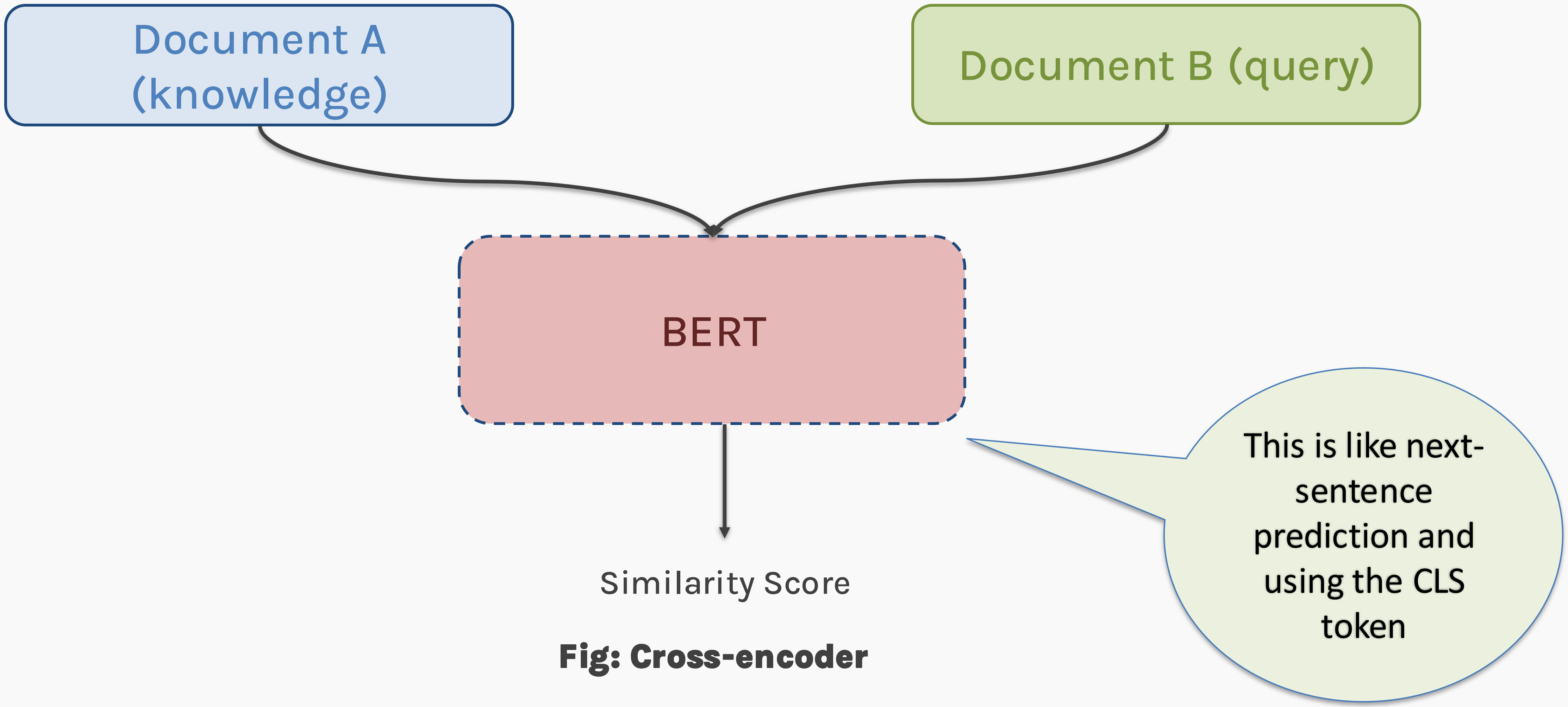


Fig: Cross-encoder

Can you guess why reranking is slower compared to retrieving?

Post-Retrieval – Re-ranking

- Unlike the naïve retrieval, the cross-encoder does not use a simple formula to compare vectors, mitigating information loss.

- We feed the document and query vectors into the cross-encoder, run it and output a single similarity score.

This leads to better results than retrieving.

The main drawback to this is, it takes time.

Thus, retrieving and reranking mitigates each other's drawbacks (information loss and time).

Thus, retrieving and reranking mitigates each other's drawbacks (information loss and time).

Outline

- Naïve RAG - Recap
- Pre-retrieval Optimization
- Retrieval Optimization
- Post-Retrieval Optimization
- **Self-RAG**
- Corrective-RAG

Self-RAG

Self-RAG

Limitations of Advanced RAG:

1. Doesn’t guarantee the relevancy of the chunk to the query.

2. No guarantee that the response from LLM using the k-chunks are related to the chunks themselves (hallucinations).

3. Doesn’t consider the possibilities where retrieval may not be necessary.

Self-RAG - Intuition

Let's take a scenario where you're taking an open book exam. How would you go about it?

A) For familiar topics, answer quickly; for unfamiliar ones, refer to the book, find relevant parts and then answer.

B) For every topic, refer to the book, find relevant section and write the answer.

Self-RAG - Intuition

Let's take a scenario where you're taking an open book exam. How would you go about it?

A) For familiar topics, answer quickly; for unfamiliar ones, refer to the book, find relevant parts and then answer.

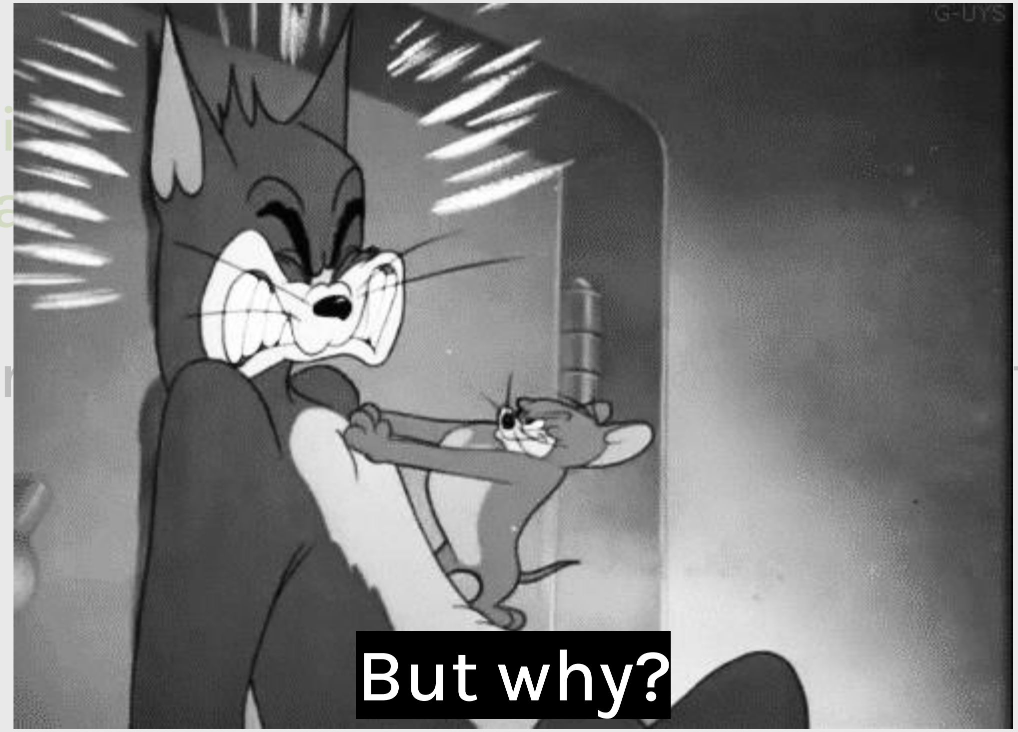
B) For every topic, refer to the book, find relevant section and write the answer.

Self-RAG - Intuition

Let's take a scenario where you're taking an open book exam. How would you go about it?

A) For familiar topi … ones, refer to the book, find releva …

B) For every topic, r … tion and write the answer.



Self-RAG - Intuition

Referring to the book even when it is not needed can lead to:

1. Slower response rate.

2. More confusion and mistakes.

3. Introduction of irrelevant or erroneous information, while scouring through the book.

Similarly, there may be times when it's not required for the RAG to retrieve documents from the vector database.

So, how do we fix this problem?

Self-RAG - Intuition

Referring to the book even when it is not needed can lead to:

1. Slower response rate.
2. More confusion and mistakes.

Self-RAG

3. Introduction of irreleva rmation, whilst scouring through the book.

Similarly, there may be times when it's not required for the RAG to retrieve documents from the vector database.

So, how do we fix this problem?

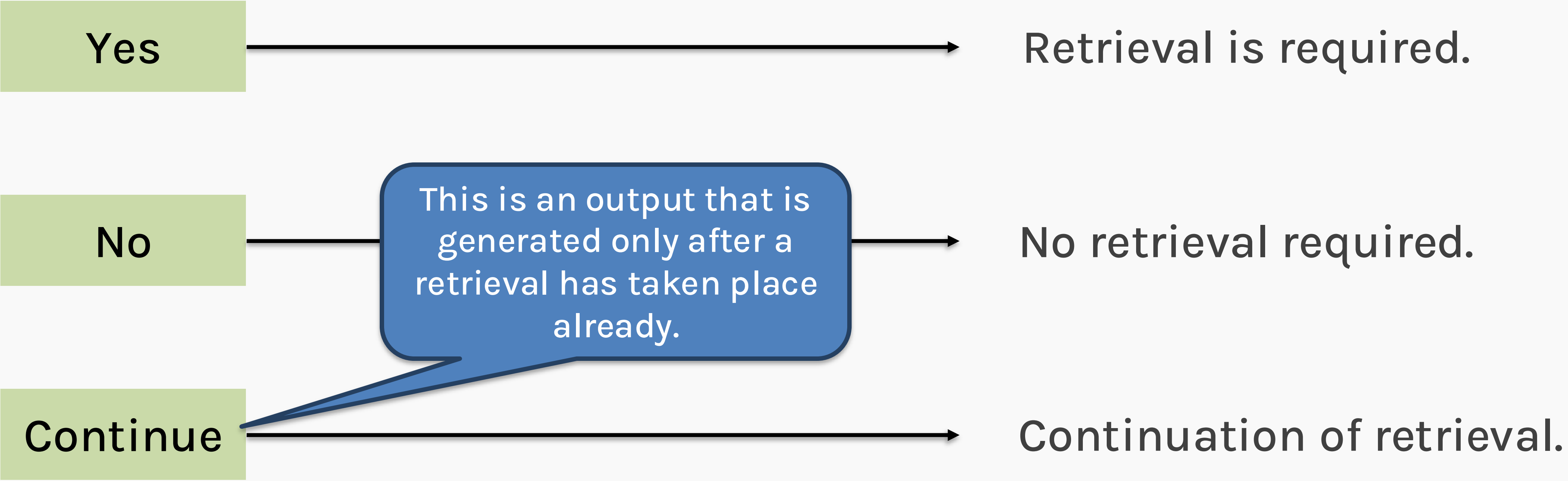
Self-RAG - Introduction

- Self-RAG is a “new” framework that controls the retrieval and generation process via reflection tokens.

- There are 2 types of reflection tokens:
    - Retrieve token: To evaluate the utility of retrieval.
    - Critique token: To evaluate the documents that have been retrieved.

Self-RAG – Retrieve token

- The retrieve token is generated by the Self-RAG to evaluate the utility of retrieval.
- It has 3 possible outputs.



Self-RAG – Critique token

- The critique token is generated by the Self-RAG to evaluate the documents that have been retrieved.

- It can be further subdivided into 3 types of tokens:

    - ISREL: Determ … ved document provides useful infor … tion to solve the qu …
    - ISSUP: Determines if the output generated is supporte… by the retrieved document.
    - ISUSE: Determines if the output generated is useful to the query.

A hallucination check

Useful basically means: Does it answer the query?

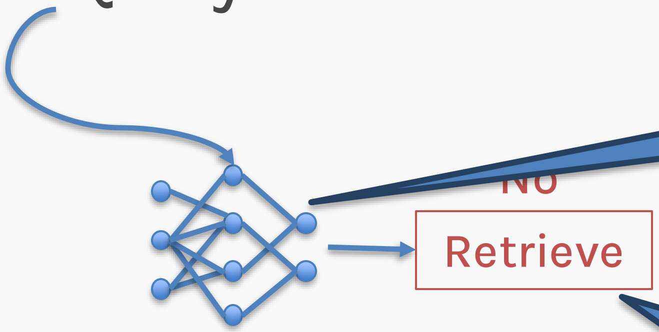
Let's look at an example to clarify these concepts!

Self-RAG – Example

Let's look at a query where the model already knows how to a

Query: Write an essay about summer vacation?

This is a fine-tuned LLM to answer this type of questions as well.



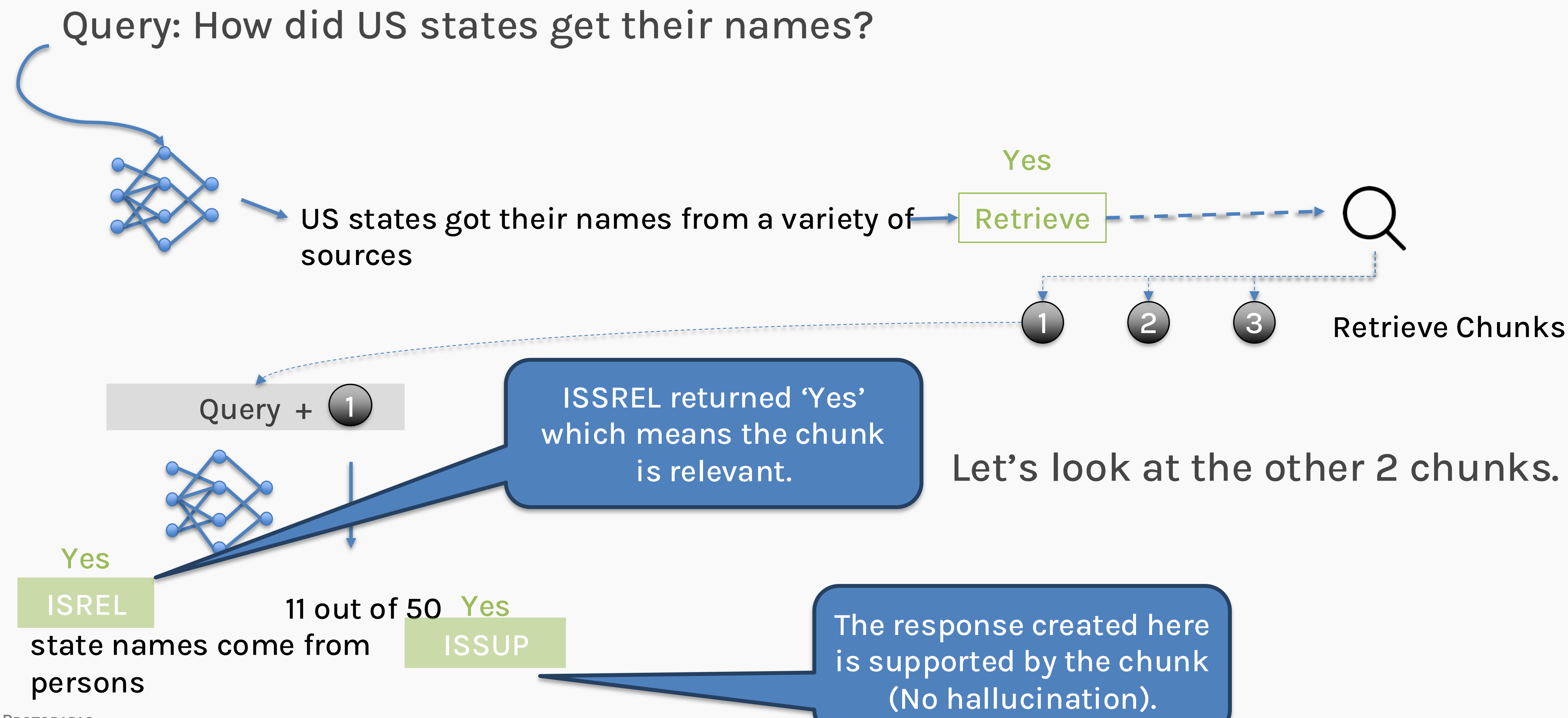
My best summer vacation is when my family and I went on a road trip along...

The retrieve token has returned a 'No', hence no retrieval is required.

Now, let's look at a case where the model may not have all the facts to answer the question.

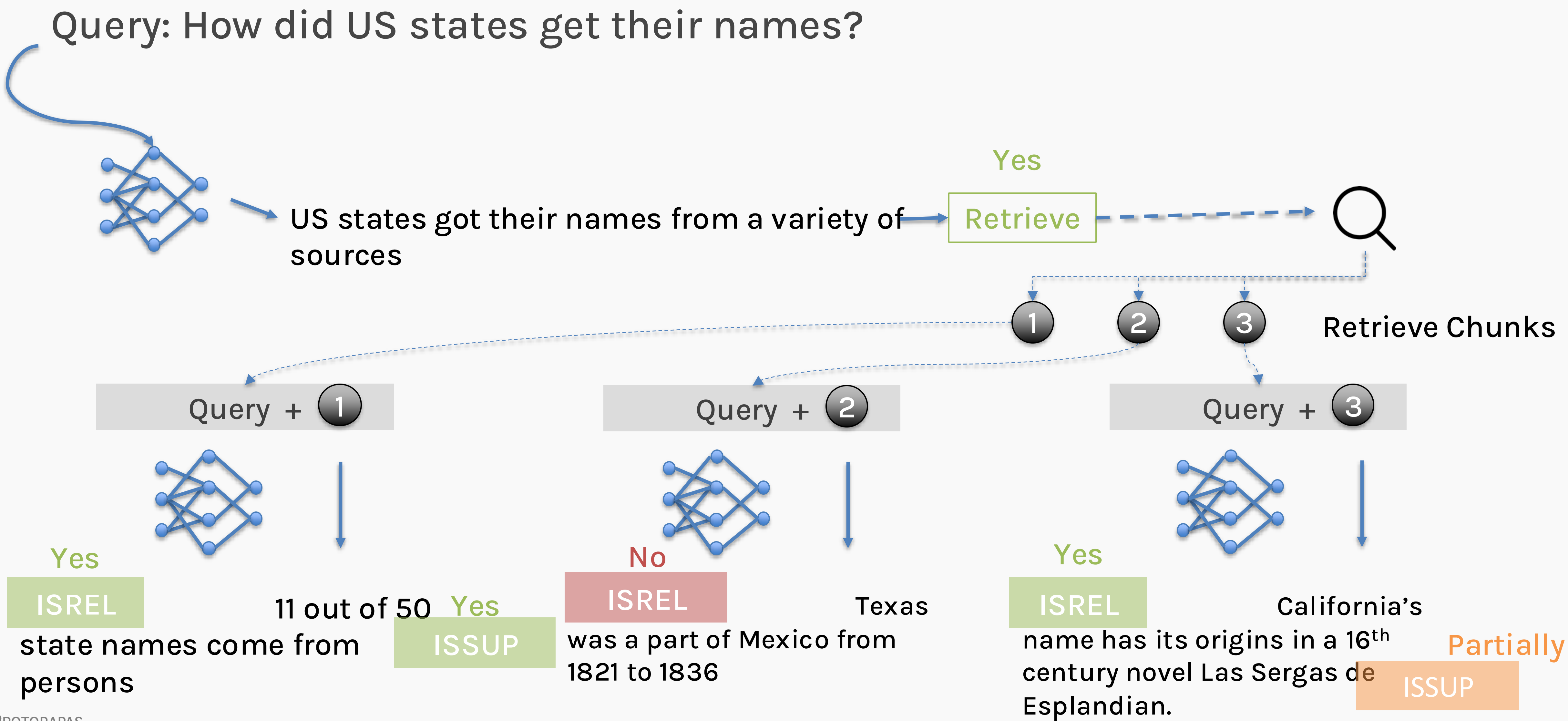
Asai et al.

Self-RAG - Example



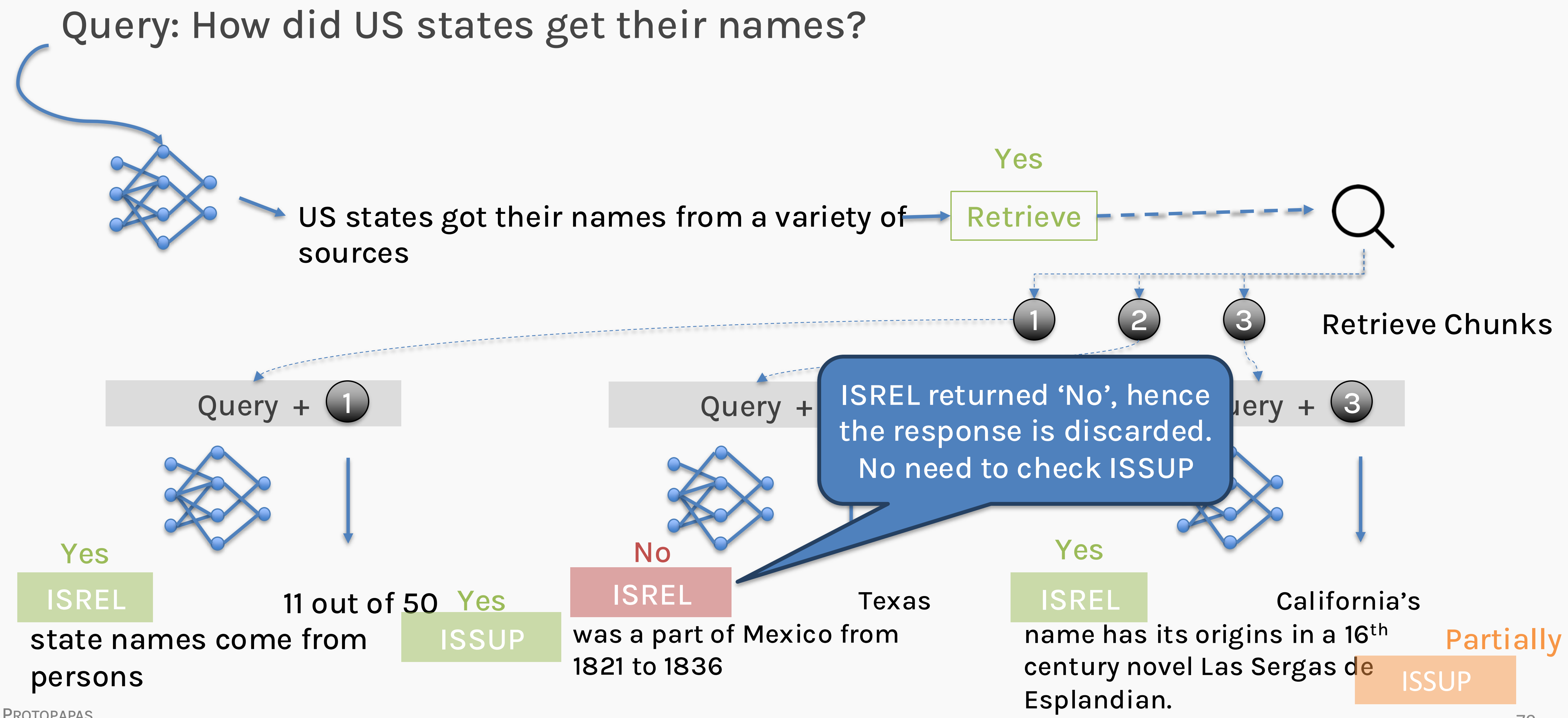
Asai et al.

Self-RAG - Example



[Asai et al.](#)

Self-RAG - Example



[Asai et al.](#)

Self-RAG - Example

Query: How did US states get their names?

Asai et al.

Self-RAG - Example

Response 1

11 out of 50 state names come from persons

ISUSE

5

Response 2

Texas was a part of Mexico from 1821 to 1836

ISUSE

1

Response 3

California's name has its origins in a 16th century novel Las Sergas de Esplandian.

ISUSE

3

The ISUSE token returns a rating of 1-5,
Where 5 is the highest rating and 1 is the lowest.

Asai et al.

Self-RAG - Example

Response 1

11 out of 50 state names come from persons

We now have a response that can be returned by the LLM.

New Response

US states got their names from a variety of Sources. 11 out of 50 state names come from persons.

The Self-RAG now checks if the created response is good enough or if more retrieval is required.

But how does Self-RAG check?

Asai et al.

Self-RAG - Example

Response 1

11 out of 50 state names come from persons

We now have a response that can be returned by the LLM.

New Response

The RETRIEVE token is used! If it returns **'Continue'**, we retrieve some more chunks.

US states got their names fr
Sources. 11 out of 50 state na
from persons.

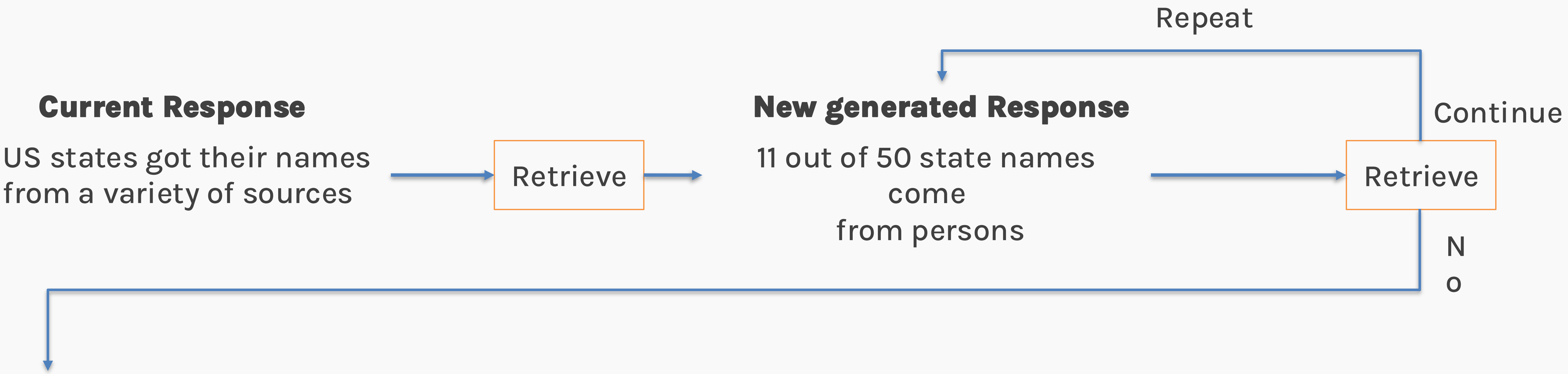
the created response is
good enough or if more retrieval is required.

But how does Self-RAG check?

[Asai et al.](#)

Self-RAG - Example

Query: How did US states get their names?



Final Response US states got their names from a variety of sources. 11 out of 50 state names come from persons. 26 states are named after Native Americans, including Utah.

Self-RAG

Type	Input	Ouput	Definition
Retrieve	(Query) or (Query, retrieved chunk, and previous segments – if any)	{yes, no, continue}	Decided if to use the retriever
IsREL	Query, Retrieved Chunk	{relevant, irrelevant}	If chunk proves useful information to solve query
IsSUP	Query, Retrieved Chunk, Current Output	{fully supported, partially supported, no support}	If current segment is supported by the chunk
IsUSE	Query, Current Output	{5, 4, 3, 2, 1}	If current output is a useful response to the query

Outline

- Naïve RAG - Recap
- Pre-retrieval Optimization
- Retrieval Optimization
- Post-Retrieval Optimization
- Self-RAG
- **Corrective-RAG**

Corrective-RAG

Corrective-RAG

When reading a book, we often come across information which is insufficient or ambiguous.

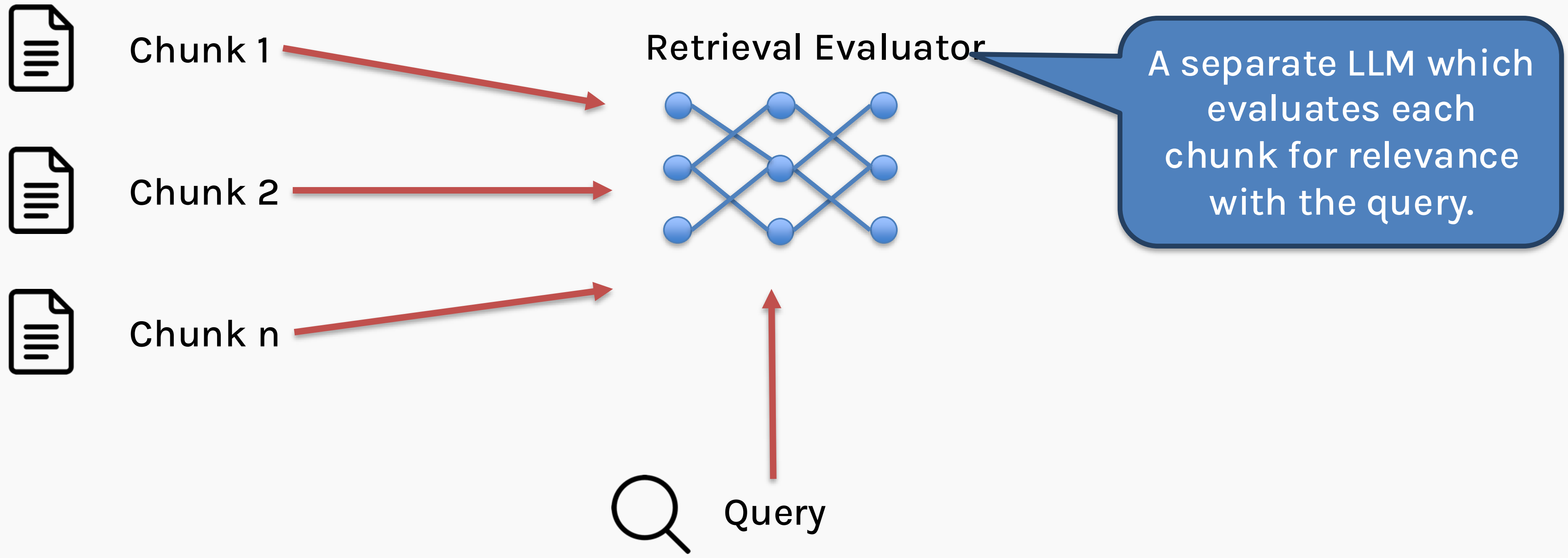
So, what do we do then?

Solution: We refer to the internet for additional details.

That’s exactly what our next variant of RAG does!

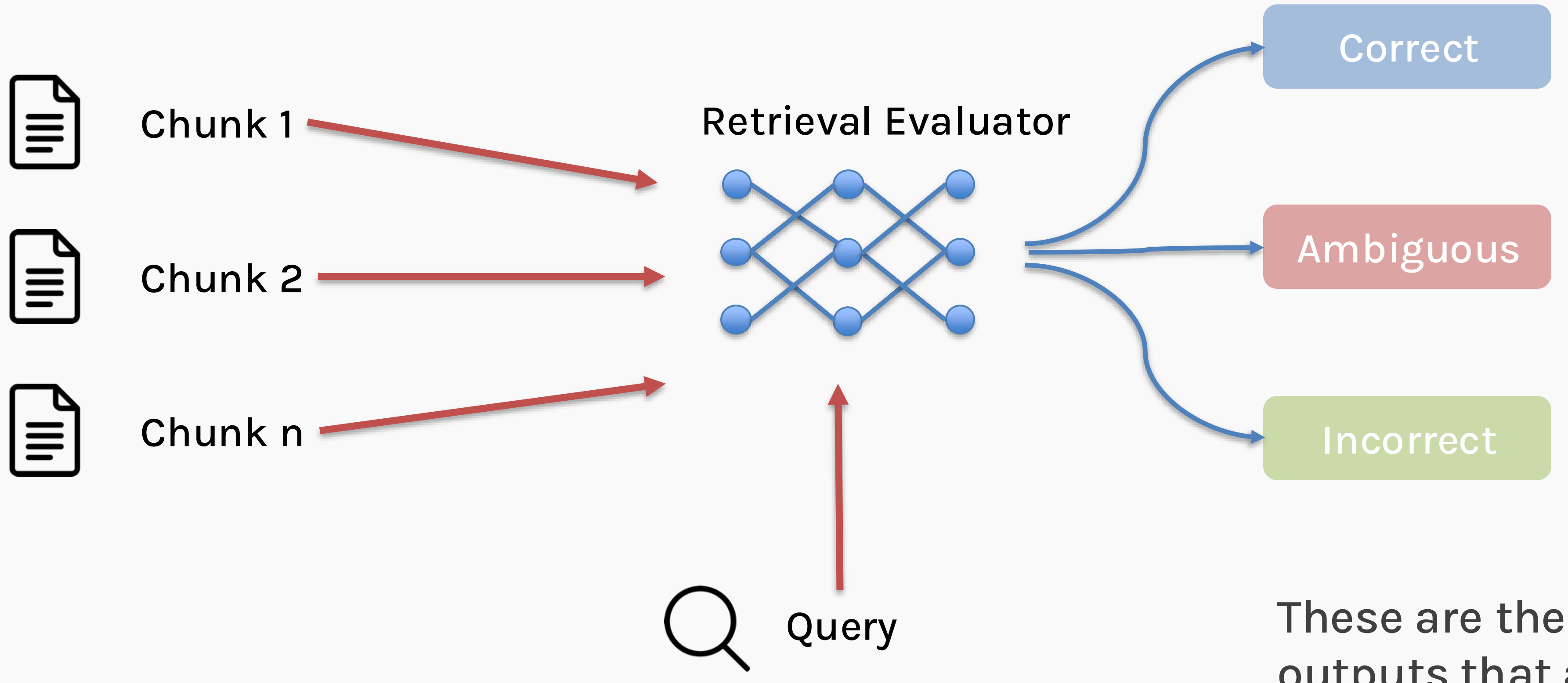
Corrective-RAG

Let's suppose these are the chunks/documents we got after we retrieve and re-rank:



Corrective-RAG

Let's suppose these are the chunks/documents we got after we retrieve and re-rank:

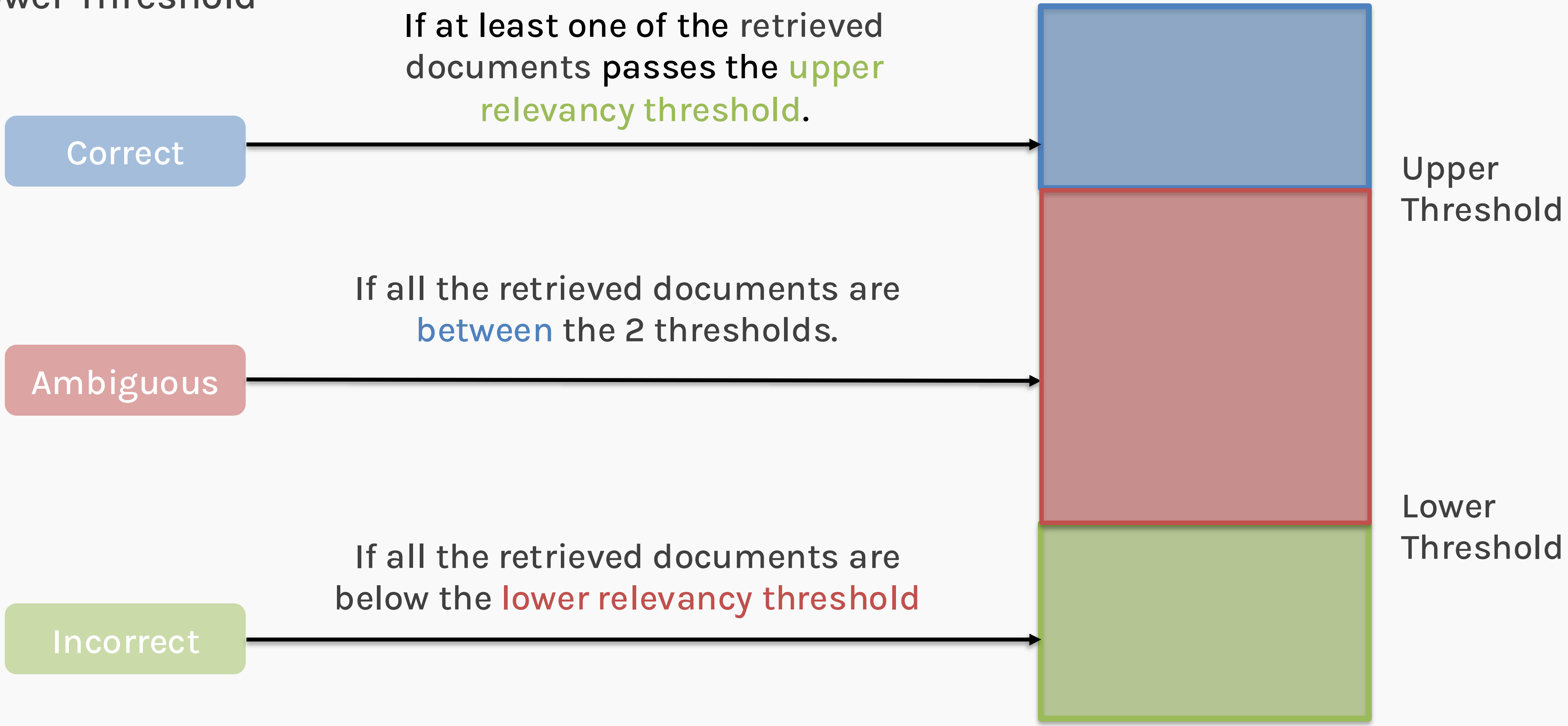


These are the 3 possible outputs that are given by the evaluator

Corrective-RAG

The 3 outputs are given based on 2 thresholds which are set beforehand.

1. Upper Threshold
2. Lower Threshold



Corrective-RAG

Correct

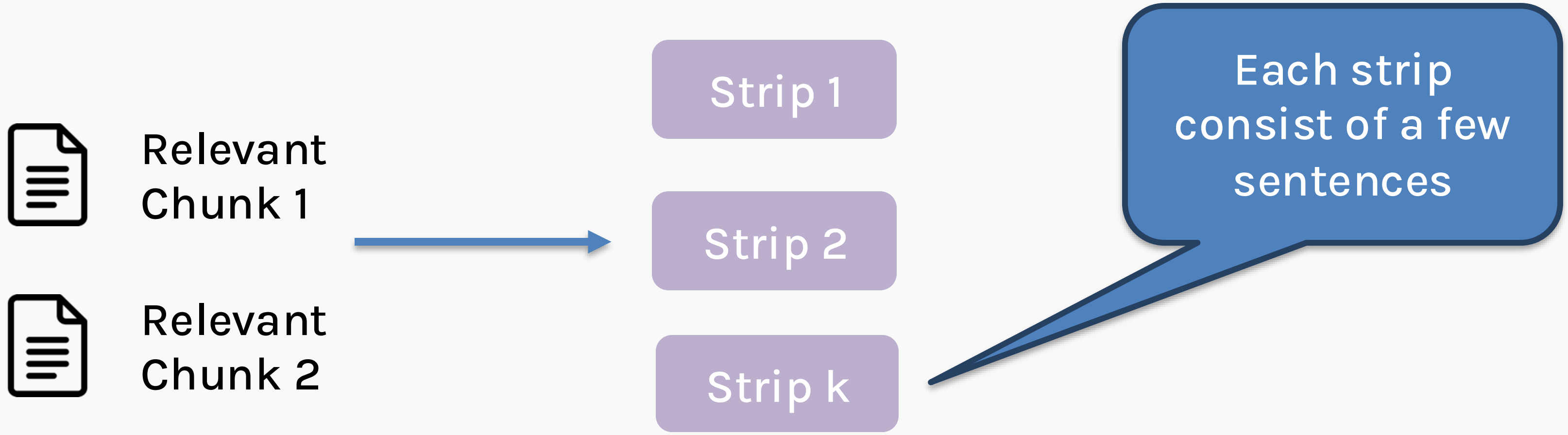
Ambiguous

Incorrect

Correct:

This is when the chunks are relevant to the query provided.

In this case, we do knowledge refinement.



Corrective-RAG

Correct

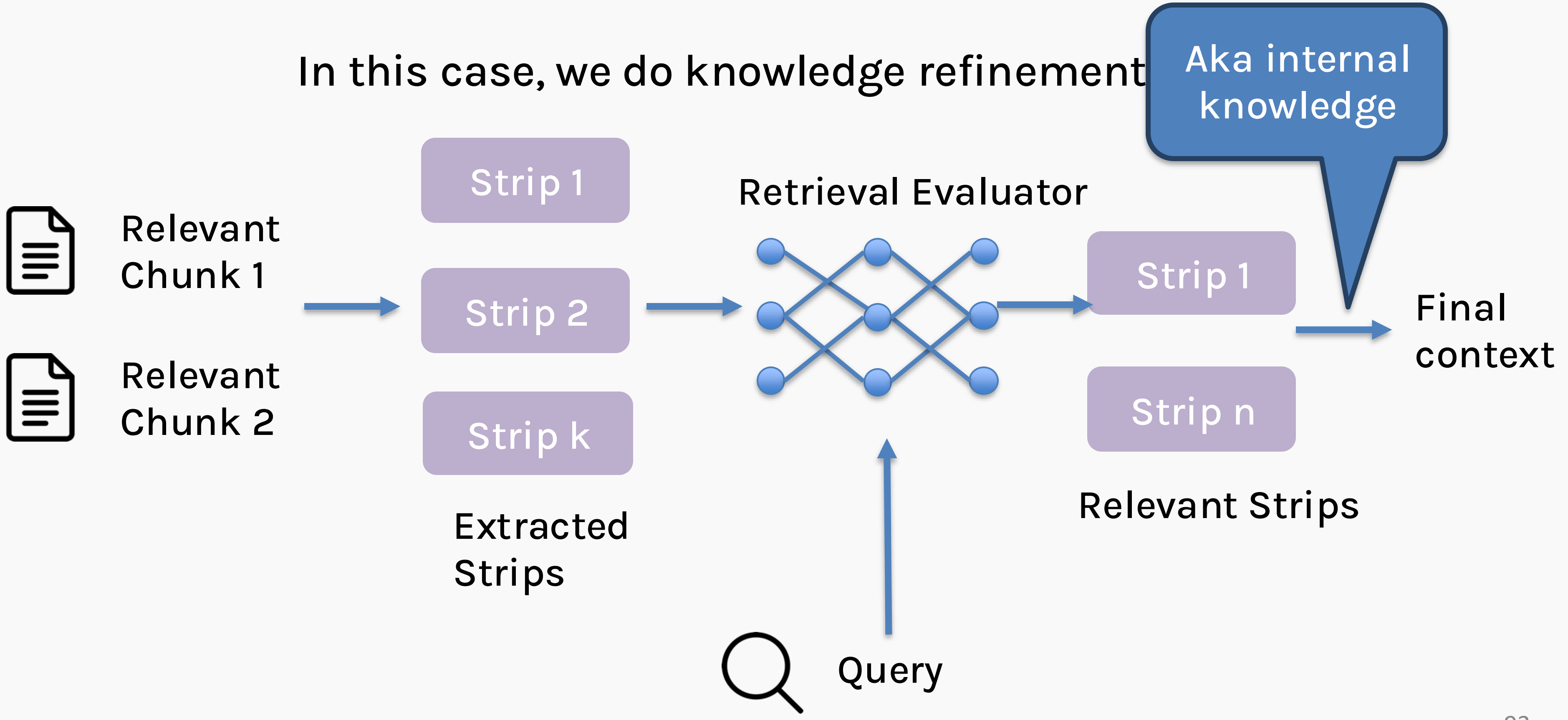
Ambiguous

Incorrect

Correct:

This is when the chunk is relevant to the query provided.

In this case, we do knowledge refinement



Corrective-RAG

Correct

Ambiguous

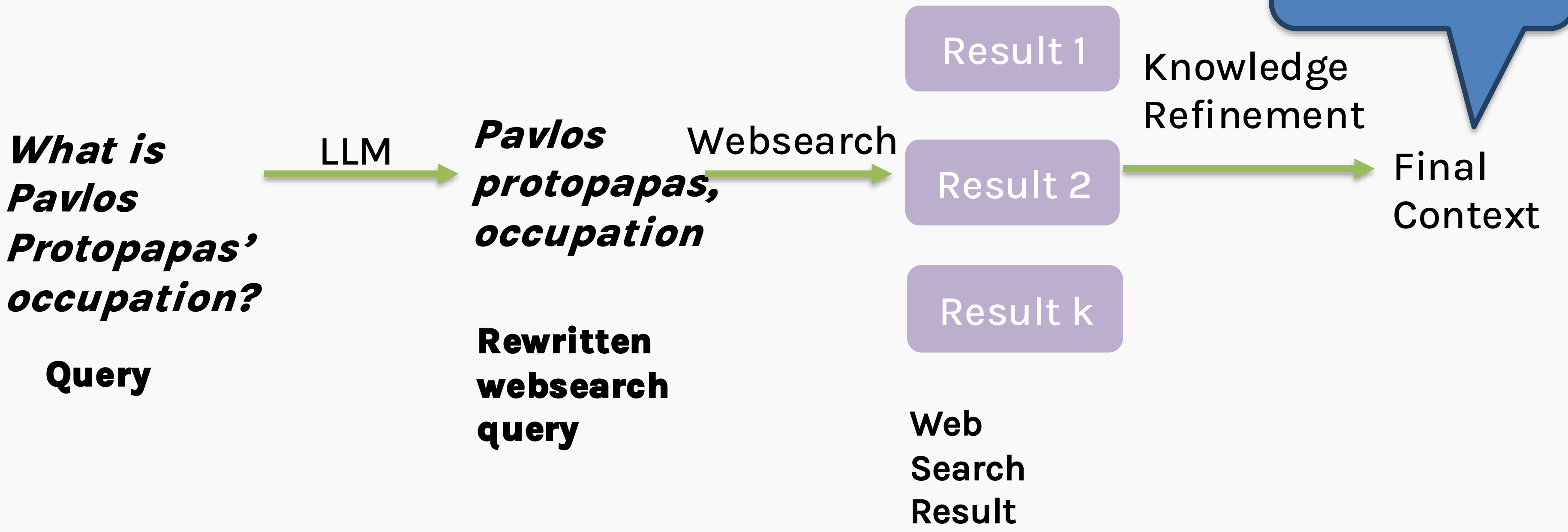
Incorrect

Incorrect:

This is when no retrieved chunk is relevant to the query.

In this case, we search the web to provide answe

Aka external knowledge



Corrective-RAG

Correct

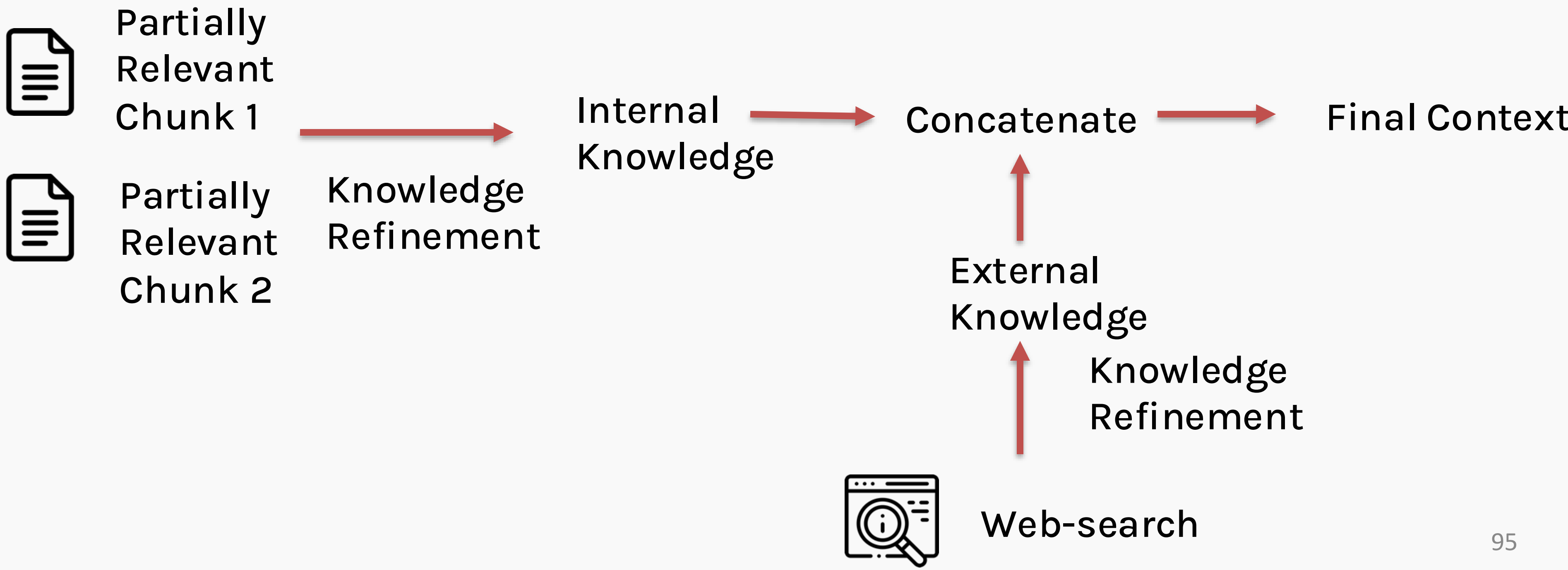
Ambiguous

Incorrect

Ambiguous:

This is when the retrieved chunks aren't correct or incorrect

In this case, we combine both the internal and external knowledge to create our final context



Corrective-RAG

- Corrective RAG is plug and play and can be combined with naive RAG, advanced RAG, and even Self-RAG.

- When we combine corrective RAG with self-RAG, we get Self-CRAG, which is the state of the art currently.

Outline

- Recap: BERT + GPT
- InstructGPT (ChatGPT)
- Prompt Engineering and Langchain
- RAG
- Advanced RAG
- Agents

LLM Applications

Let’s look at some of the LLM applications using Agentic Workflow (Agents)

AI-chatbot-powered research and conversational search engine

perplexity

LLaMA GUI

FreedomGPT

ChatLLaMA

to ChatGPT (with large context windows)

AutoGPT

Microsoft

Task Based Automated GPT / Agents

OpenAssistant

chat

Open Source Chatbot

February

March

April

30th November 2022

2023

LLM Applications

AI-chatbot-powered research and conversational search engine

perplexity

LLaMA GUI

FreedomGPT

Claude 2 AI

Meta

ChatLLaMA

Anthropic's answer to ChatGPT (with large context windows)

AutoGPT

Microsoft Jarvis

AgentGPT

Task Based Automated GPT / Agents

OpenAssistant

huggingchat

Open Source Chatbot

February

March

April

30th November 2022

2023

LLM Applications

LLaMA GUI

AI-chatbot-powered research and conversational search engine

FreedomGPT

perplexity

AutoGPT

OpenAssistant

Microsoft

Here we define 'LLM Applications' as any interface that makes accessing LLMs easier. Sometimes also called 'LLM tools'

to ChatGPT (with large context windows)

Task Based Automated GPT / Agents

Open Source Chatbot

February

March

April

30th November 2022

2023

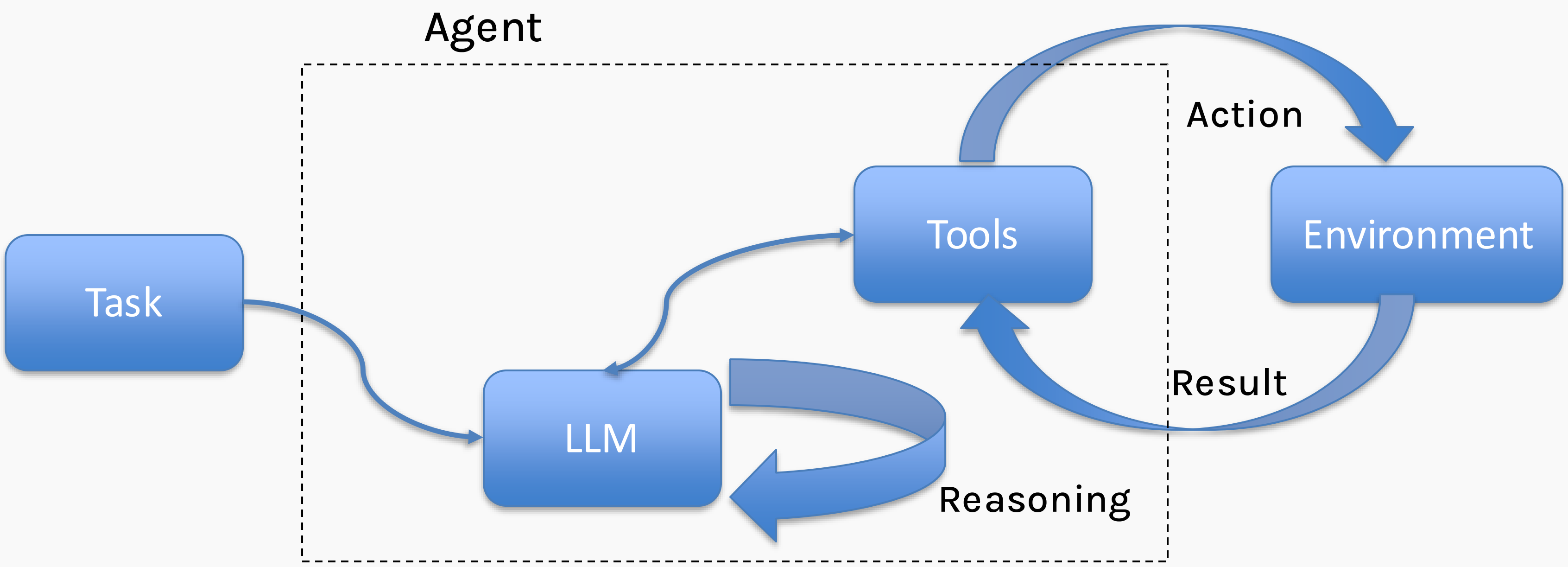
How do we define an ‘Agent’/Agentic Workflow?

“While there isn’t a widely accepted definition for LLM-powered agents, they can be described as a system that can use an LLM to reason through a problem, create a plan to solve the problem, and execute the plan with the help of a set of tools.”

Source: Nvidia

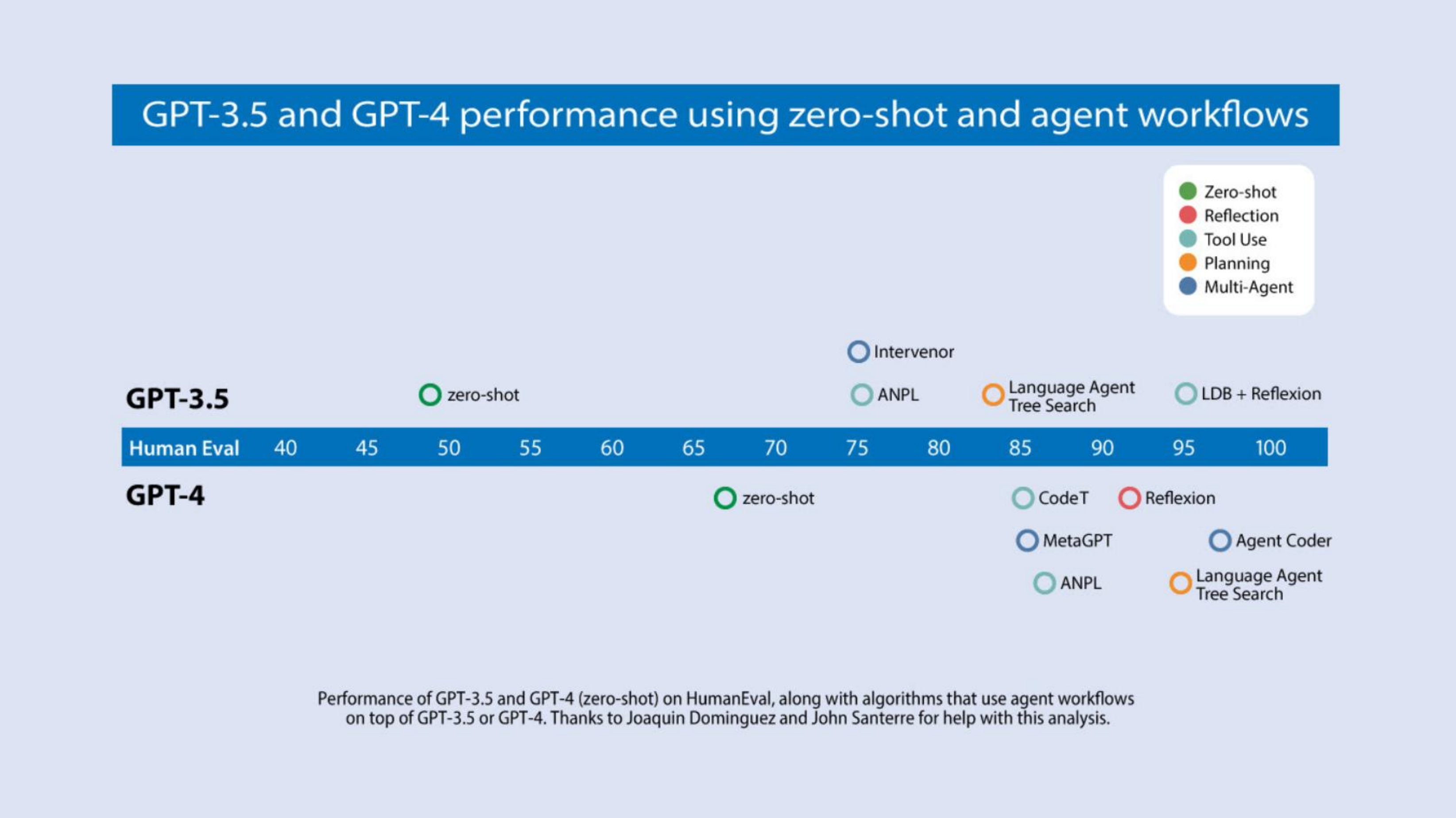
Agentic Workflow

- In other words, an agentic workflow is any multi-step process that iteratively instructs large language models to complete complex tasks.



Agentic Workflow

- In other words, an agentic workflow is any multi-step process that iteratively instructs large language models to complete complex tasks.



[Source: DeepLearningAI]

Agentic Workflow

- For ex., instead of a single prompt asking for insights from a .csv, with a workflow can allow us to guide a model to 'act like a data scientist' and work iteratively:

Streamlined Data Analysis Example:

1. **Initial Review:** Briefly assess the dataset's structure and main components.
2. **Hypothesize:** Formulate initial theories based on quick observations.
3. **Query Data:** Execute targeted data explorations, like filtering or aggregations.
4. **Draft Analysis:** Create a basic analysis report.
5. **Review:** Check the draft for logical flaws or missed insights.
6. **Refine:** Update the analysis, correcting or enhancing findings.
7. **Finalize Report:** Produce the detailed, final version of the analysis.

Agentic Workflow: Design Patterns

- According to Andrew Ng, these frameworks can prove useful to build such workflows:

Reflection: The LLM examines its own work to come up with ways to improve it.

Tool Use: The LLM is given tools such as web search, code execution, or any other function to help it gather information, take action, or process data.

Planning: The LLM comes up with, and executes, a multistep plan to achieve a goal (for example, writing an outline for an essay, then doing online research, then writing a draft, and so on).

Multi-agent collaboration: More than one AI agent work together, splitting up tasks and discussing and debating ideas, to come up with better solutions than a single agent would.

Tutorial 9: Cheese newsletter generation

In this demo, we’ll create a newsletter for Formaggio.me, highlighting the best cheese sales around the Boston area!

But here’s the twist: we won’t be manually searching the web, summarizing deals, or crafting the newsletter ourselves. Instead, we’ll let an agent handle the heavy lifting for us—searching, curating, and delivering the perfect newsletter automatically.

https://colab.research.google.com/drive/1UVn3L6KQgsrVLnLRaMVbpV3VJr_i5MLW?usp=sharing


Prof. Protopapas
Students