

# LECTURE 9: ADVANCED USES OF LLMS

---

Creating Business Value with Generative AI  
Fall 2025



AARHUS  
BSS  
DEPARTMENT OF MANAGEMENT  
AARHUS UNIVERSITY

29. October 2025

Magnus Bender  
Assistant Professor



# WHY THIS LECTURE?

---



DEPARTMENT OF MANAGEMENT  
AARHUS UNIVERSITY

29. October 2025

Magnus Bender  
Assistant Professor

2



# WHY THIS LECTURE?

---

- To create *value* using LLMs we need more than just a *new* interface for the OpenAI API

# WHY THIS LECTURE?

---

- To create *value* using LLMs we need more than just a *new* interface for the OpenAI API
  - Add specific rules → Prompts, Structured outputs via pydantic
  - Integrate in a Python program and a process

# WHY THIS LECTURE?

---

- To create *value* using LLMs we need more than just a *new* interface for the OpenAI API
  - Add specific rules → Prompts, Structured outputs via pydantic
  - Integrate in a Python program and a process
  - Add specific data → ?

# WHY THIS LECTURE?

---

- To create *value* using LLMs we need more than just a *new* interface for the OpenAI API
  - Add specific rules → Prompts, Structured outputs via pydantic
  - Integrate in a Python program and a process
  - Add specific data → ?
  - Allow access to specific tools (Python functions) via GPT (the OpenAI API) → ?

# WHY THIS LECTURE?

---

- To create *value* using LLMs we need more than just a *new* interface for the OpenAI API
  - Add specific rules → Prompts, Structured outputs via pydantic
  - Integrate in a Python program and a process
  - Add specific data → ?
  - Allow access to specific tools (Python functions) via GPT (the OpenAI API) → ?
- Demonstrate for your projects how to

# WHY THIS LECTURE?

---

- To create *value* using LLMs we need more than just a *new* interface for the OpenAI API
  - Add specific rules → Prompts, Structured outputs via pydantic
  - Integrate in a Python program and a process
  - Add specific data → ?
  - Allow access to specific tools (Python functions) via GPT (the OpenAI API) → ?
- Demonstrate for your projects how to
  - ... use a slightly bigger data source, i.e.,
  - ... incorporate use-case specific data, information, FAQs

# WHY THIS LECTURE?

---

- To create *value* using LLMs we need more than just a *new* interface for the OpenAI API
  - Add specific rules → Prompts, Structured outputs via pydantic
  - Integrate in a Python program and a process
  - Add specific data → ?
  - Allow access to specific tools (Python functions) via GPT (the OpenAI API) → ?
- Demonstrate for your projects how to
  - ... use a slightly bigger data source, i.e.,
  - ... incorporate use-case specific data, information, FAQs
- Get more functionality in Python → Python packages

# PROJECT & PROPOSALS

---



DEPARTMENT OF MANAGEMENT  
AARHUS UNIVERSITY

29. October 2025

Magnus Bender  
Assistant Professor

3



# PROJECT & PROPOSALS

---

- Feedback process
  - We use the used GPT-based tool (presented last lecture)

# PROJECT & PROPOSALS

---

- Feedback process
  - We use the used GPT-based tool (presented last lecture)
  - Feedback is based on *our human judgement* and a final check of the generated feedback

# PROJECT & PROPOSALS

---

- Feedback process
  - We use the used GPT-based tool (presented last lecture)
  - Feedback is based on *our human judgement* and a final check of the generated feedback
- Recurring topics
  - „Train“ an the LLM → use RAG, we cannot change the LLMs on OpenAI's servers

# PROJECT & PROPOSALS

---

- Feedback process
  - We use the used GPT-based tool (presented last lecture)
  - Feedback is based on *our human judgement* and a final check of the generated feedback
- Recurring topics
  - „Train“ an the LLM → use RAG, we cannot change the LLMs on OpenAI's servers
  - Think about validation and evaluation

# PROJECT & PROPOSALS

---

- Feedback process
  - We use the used GPT-based tool (presented last lecture)
  - Feedback is based on *our human judgement* and a final check of the generated feedback
- Recurring topics
  - „Train“ an the LLM → use RAG, we cannot change the LLMs on OpenAI's servers
  - Think about validation and evaluation
- General comments

# PROJECT & PROPOSALS

---

- Feedback process
  - We use the used GPT-based tool (presented last lecture)
  - Feedback is based on *our human judgement* and a final check of the generated feedback
- Recurring topics
  - „Train“ an the LLM → use RAG, we cannot change the LLMs on OpenAI's servers
  - Think about validation and evaluation
- General comments
  - *To-be* and *as-is* as diagrams with details

# PROJECT & PROPOSALS

---

- Feedback process
  - We use the used GPT-based tool (presented last lecture)
  - Feedback is based on *our human judgement* and a final check of the generated feedback
- Recurring topics
  - „Train“ an the LLM → use RAG, we cannot change the LLMs on OpenAI's servers
  - Think about validation and evaluation
- General comments
  - *To-be* and *as-is* as diagrams with details
  - Two pages goal

# PROJECT & PROPOSALS

---

- Feedback process
  - We use the used GPT-based tool (presented last lecture)
  - Feedback is based on *our human judgement* and a final check of the generated feedback
- Recurring topics
  - „Train“ an the LLM → use RAG, we cannot change the LLMs on OpenAI's servers
  - Think about validation and evaluation
- General comments
  - *To-be* and *as-is* as diagrams with details
  - Two pages goal
  - Specific examples and use-cases

# PROJECT & PROPOSALS

---

- Feedback process
  - We use the used GPT-based tool (presented last lecture)
  - Feedback is based on *our human judgement* and a final check of the generated feedback
- Recurring topics
  - „Train“ an the LLM → use RAG, we cannot change the LLMs on OpenAI's servers
  - Think about validation and evaluation
- General comments
  - *To-be* and *as-is* as diagrams with details
  - Two pages goal
  - Specific examples and use-cases
  - Generate *value* with GenAI, not „use ChatGPT to do x“

# PROJECT & PROPOSALS

---

Please ask us, if feedback  
is unclear to you or if  
there are open questions!

- Feedback process
  - We use the used GPT-based tool (presented last lecture)
  - Feedback is based on *our human judgement* and a final check of the generated feedback
- Recurring topics
  - „Train“ an the LLM → use RAG, we cannot change the LLMs on OpenAI's servers
  - Think about validation and evaluation
- General comments
  - *To-be* and *as-is* as diagrams with details
  - Two pages goal
  - Specific examples and use-cases
  - Generate *value* with GenAI, not „use ChatGPT to do x“

# AGENDA FOR TODAY

---



DEPARTMENT OF MANAGEMENT  
AARHUS UNIVERSITY

29. October 2025

Magnus Bender  
Assistant Professor

4



# AGENDA FOR TODAY

---

## A. Advanced use of LLMs

# AGENDA FOR TODAY

---

- A. Advanced use of LLMs
  - Retrieval Augmentes Generation (RAG)

# AGENDA FOR TODAY

---

## A. Advanced use of LLMs

- Retrieval Augmentes Generation (RAG)
  - Embedding of text
  - Retrieval based on embeddings
  - Relevant parameters for RAG

# AGENDA FOR TODAY

---

## A. Advanced use of LLMs

- Retrieval Augmentes Generation (RAG)
  - Embedding of text
  - Retrieval based on embeddings
  - Relevant parameters for RAG
- Function calling
  - Give GPT access to tools, i.e., Python functions

# AGENDA FOR TODAY

---

## A. Advanced use of LLMs

- Retrieval Augmentes Generation (RAG)
  - Embedding of text
  - Retrieval based on embeddings
  - Relevant parameters for RAG
- Function calling
  - Give GPT access to tools, i.e., Python functions

## B. Python packages and how to manage them

- How to get the textual content from a PDF file

# RETRIEVAL AUGMENTED GENERATION

---

... the basic RAG idea



AARHUS  
BSS  
DEPARTMENT OF MANAGEMENT  
AARHUS UNIVERSITY

29. October 2025

Magnus Bender  
Assistant Professor



# ADDING DATA TO AN LLM: TRAINING

---

- Train the model on the data

# ADDING DATA TO AN LLM: TRAINING

---

- Train the model on the data
- So-called *fine-tuning*



General LLM

# ADDING DATA TO AN LLM: TRAINING

---

- Train the model on the data
- So-called *fine-tuning*
  - Creates a new personalized model



General LLM

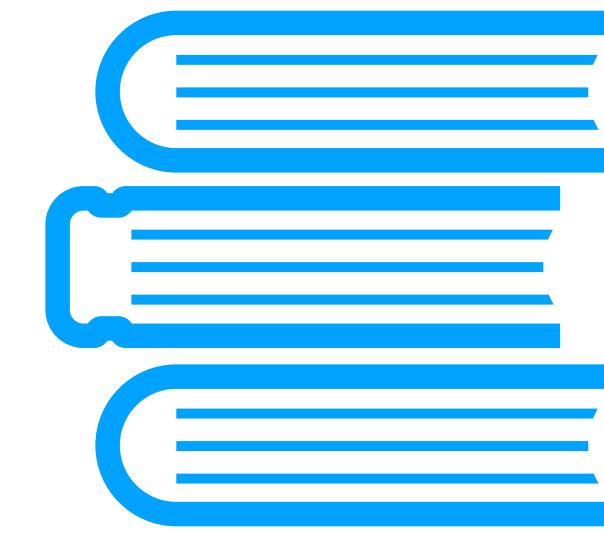


Specialized model for specific dataset

# ADDING DATA TO AN LLM: TRAINING

---

- Train the model on the data
- So-called *fine-tuning*
  - Creates a new personalized model
  - Requires larger amount of training data



Huge specific dataset



General LLM

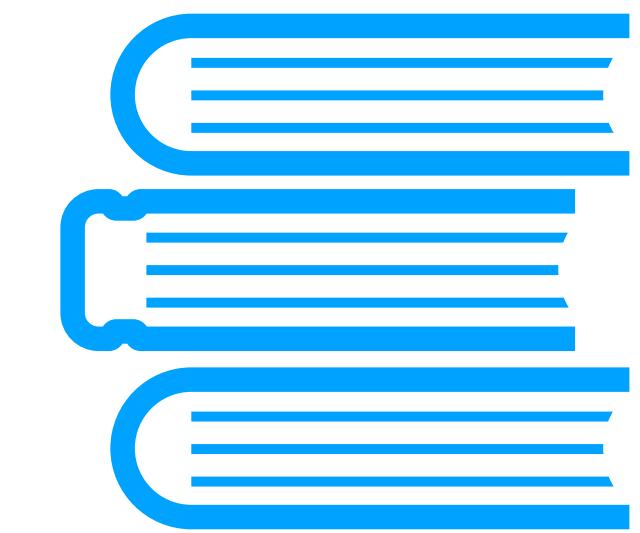


Specialized model for specific dataset

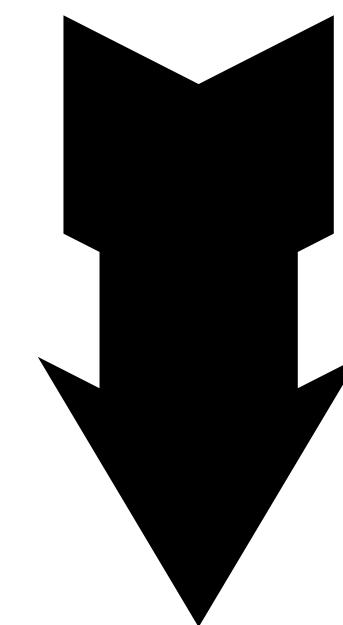
# ADDING DATA TO AN LLM: TRAINING

---

- Train the model on the data
- So-called *fine-tuning*
  - Creates a new personalized model
  - Requires larger amount of training data
  - Requires computational resources
  - Difficult to check if data really went into the model → only updated weights



Huge specific dataset



General LLM

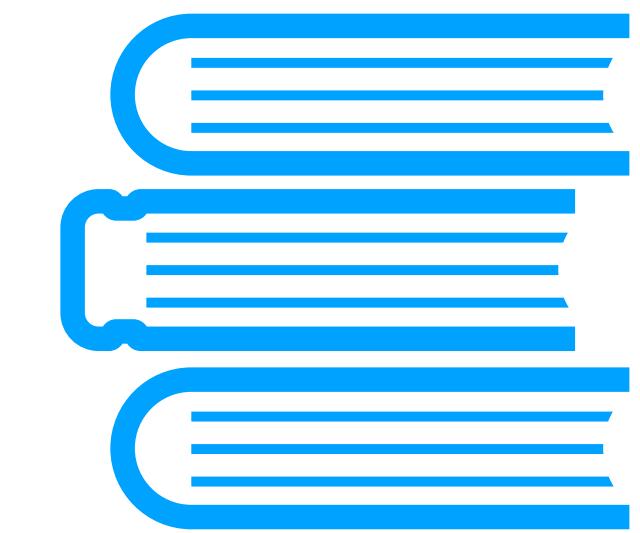


Specialized model for specific dataset

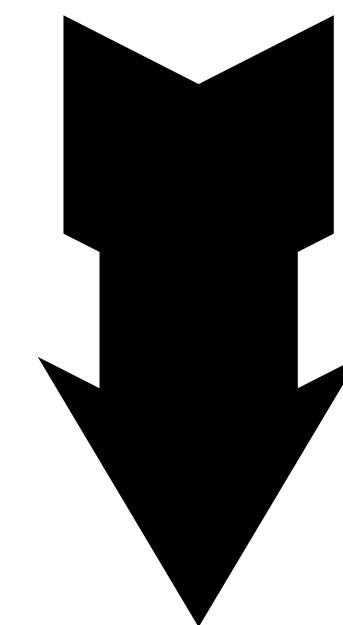
# ADDING DATA TO AN LLM: TRAINING

---

- Train the model on the data
- So-called *fine-tuning*
  - Creates a new personalized model
  - Requires larger amount of training data
  - Requires computational resources
  - Difficult to check if data really went into the model → only updated weights
  - Not suitable for our use-cases:
    - Models on OpenAI API cannot be fine-tuned easily
    - Stay with *general LLMs*



Huge specific dataset



Specialized model for specific dataset



General LLM

# ADDING DATA TO AN LLM: AUGMENTED GENERATION

---

- Add relevant data to the prompt

# ADDING DATA TO AN LLM: AUGMENTED GENERATION

---



Specific document

- Add relevant data to the prompt
  - Augment the models context with a specific document

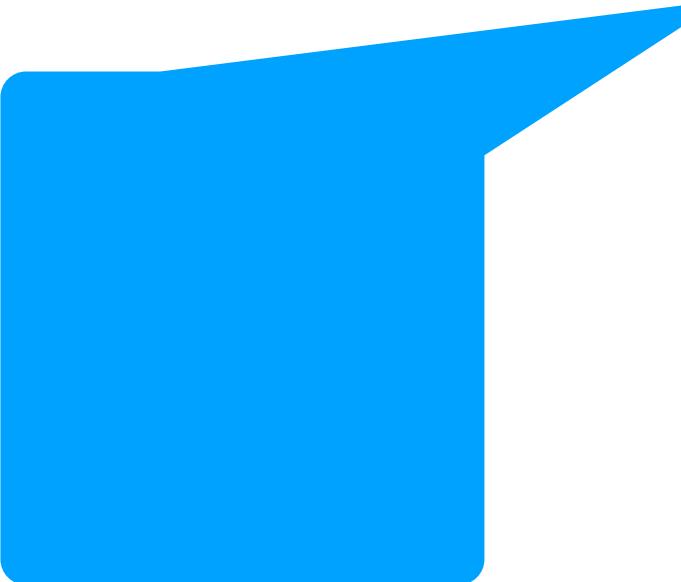
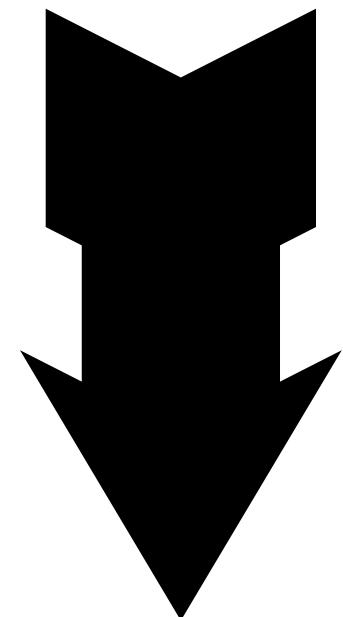
# ADDING DATA TO AN LLM: AUGMENTED GENERATION

---

- Add relevant data to the prompt
  - Augment the models context with a specific document
  - Let LLM generate the response only based on the context (information) in the prompt (e.g., question)



Specific document

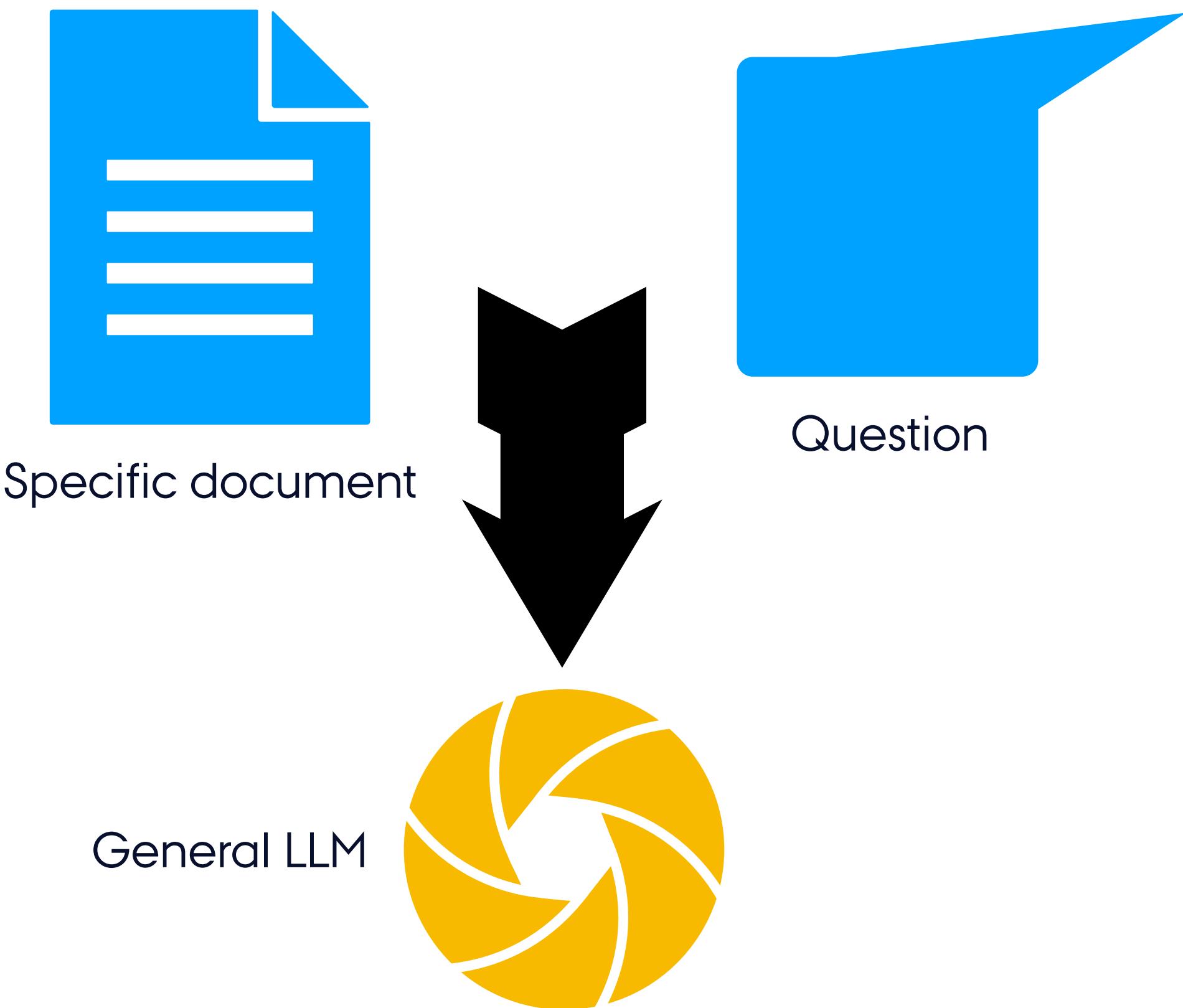


Question

# ADDING DATA TO AN LLM: AUGMENTED GENERATION

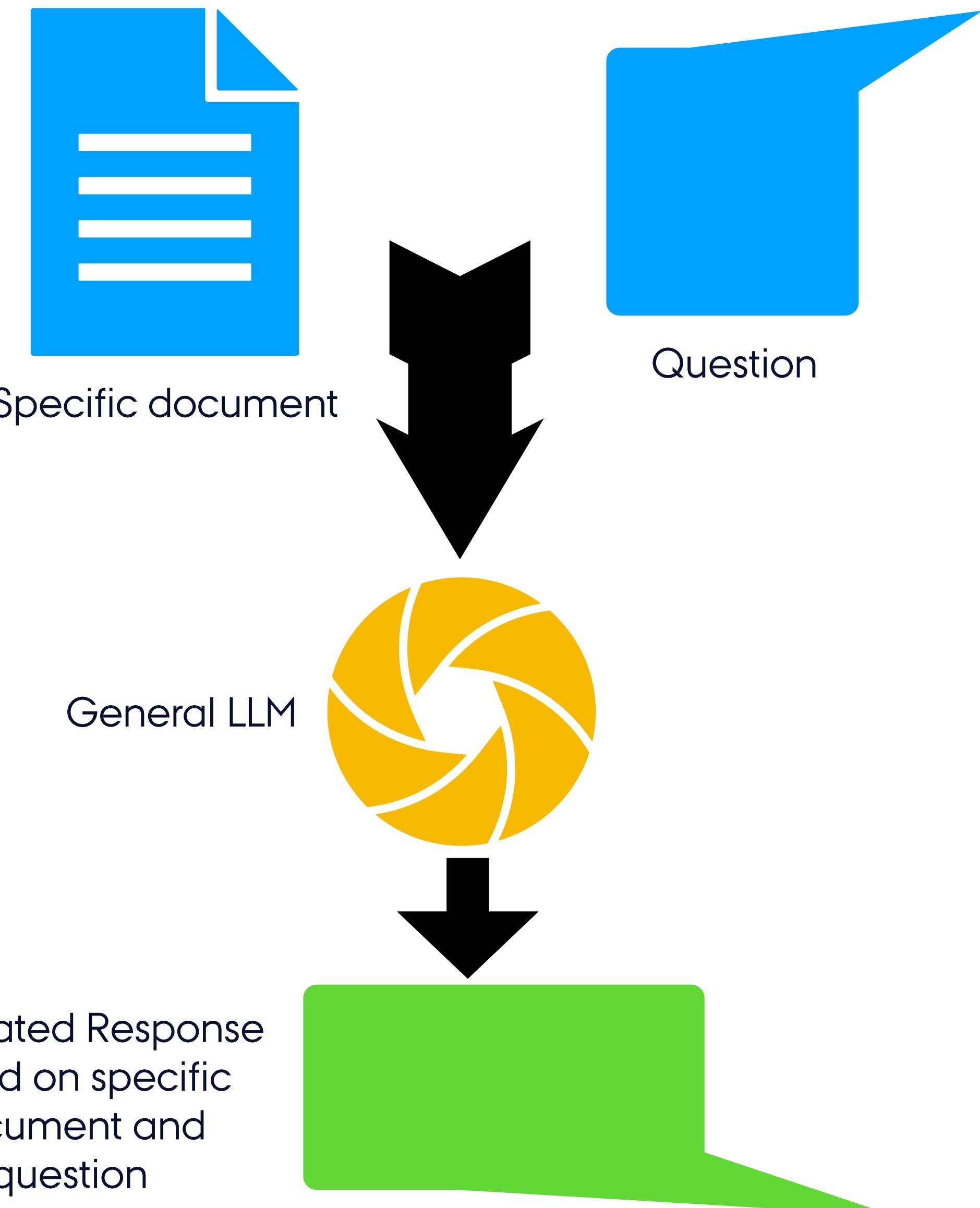
---

- Add relevant data to the prompt
  - Augment the models context with a specific document
  - Let LLM generate the response only based on the context (information) in the prompt (e.g., question)
  - Uses a *general LLM*



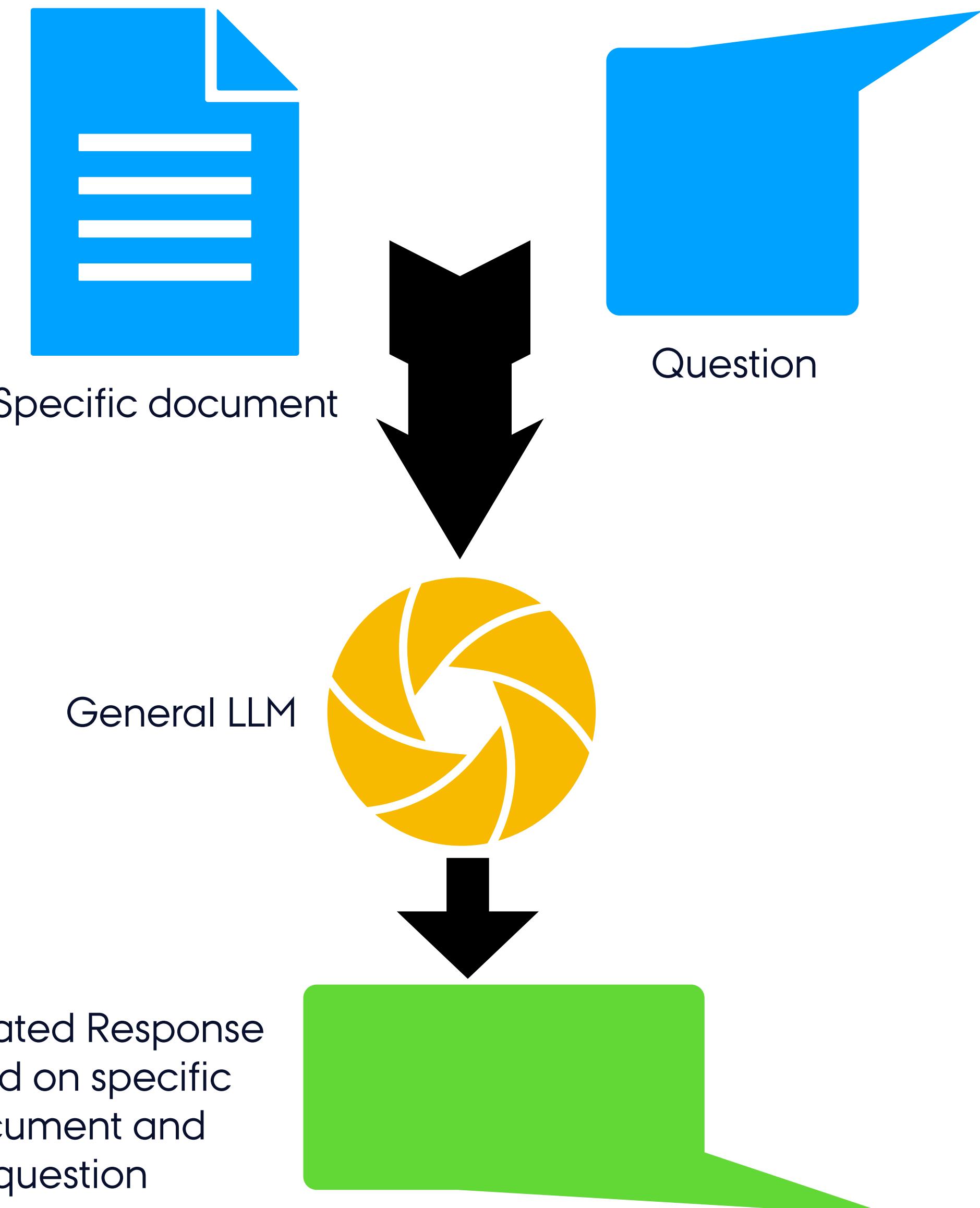
# ADDING DATA TO AN LLM: AUGMENTED GENERATION

- Add relevant data to the prompt
  - Augment the models context with a specific document
  - Let LLM generate the response only based on the context (information) in the prompt (e.g., question)
  - Uses a *general LLM*



# ADDING DATA TO AN LLM: AUGMENTED GENERATION

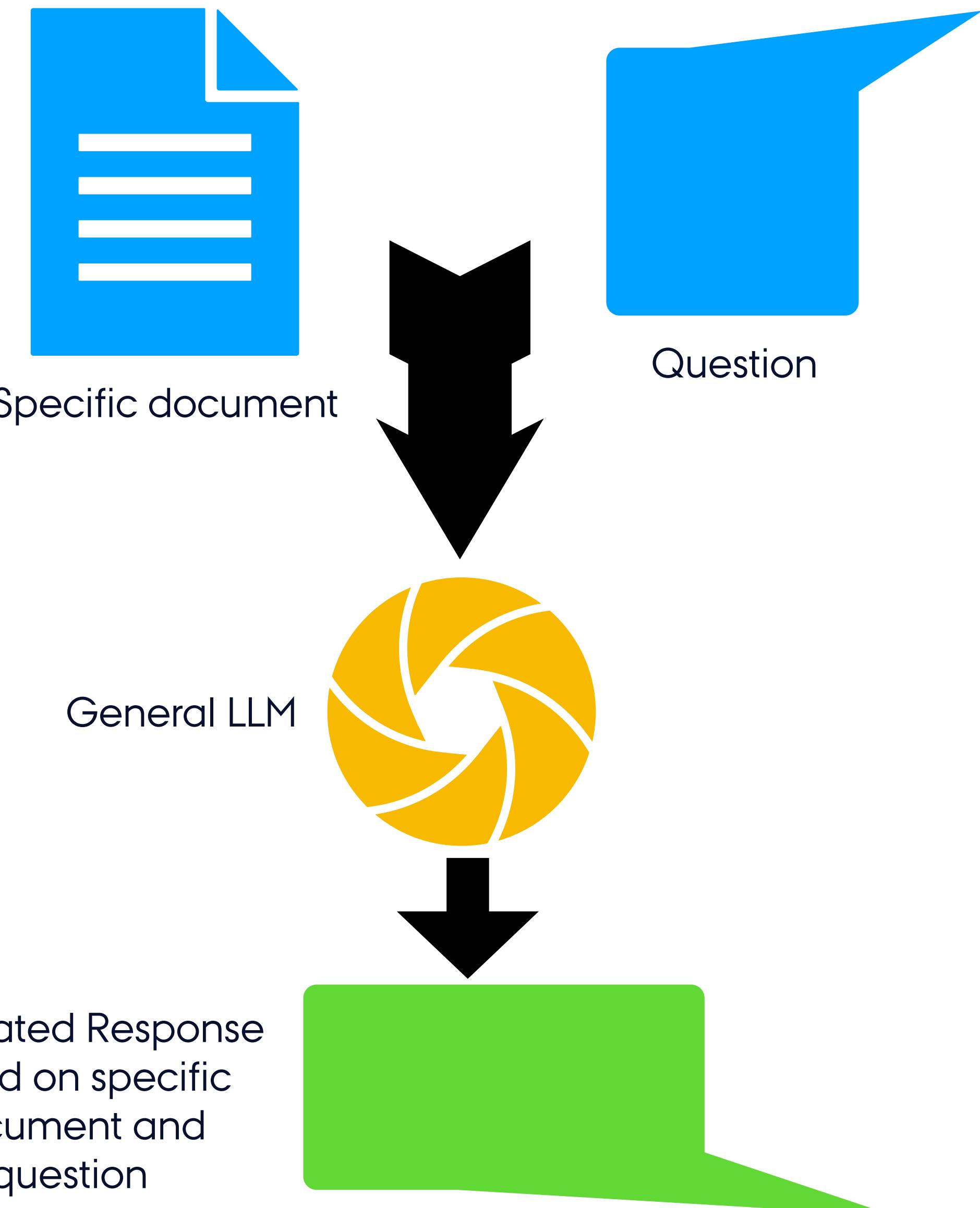
- Add relevant data to the prompt
  - Augment the models context with a specific document
  - Let LLM generate the response only based on the context (information) in the prompt (e.g., question)
  - Uses a *general LLM*
  - No training data needed
  - No computational resources for *training*, but larger context to process by LLM



# ADDING DATA TO AN LLM: AUGMENTED GENERATION

---

- Add relevant data to the prompt
  - Augment the models context with a specific document
  - Let LLM generate the response only based on the context (information) in the prompt (e.g., question)
  - Uses a *general LLM*
  - No training data needed
  - No computational resources for *training*, but larger context to process by LLM
  - Not suitable for large documents
    - Context of LLMs is limited



# ADDING DATA TO AN LLM: AUGMENTED GENERATION

---

- Example:

# ADDING DATA TO AN LLM: AUGMENTED GENERATION

---

- Example:
  - Asking ChatGPT about code:
    1. Python code
    2. Question

# ADDING DATA TO AUGMENTED GE

- Example:
  - Asking ChatGPT about code:

1. Python code
2. Question

```
python
1 v with ui.row():
2         _switch_augment_question = ui.switch('Augment question')
3         _switch_augment_dataset = ui.switch('Augment context', value=True)
4         _choose_top_k = ui.number("Number of chunks", value=10, min=1, max=100)
5         _choose_min_sim = ui.number("Minimum similarity", value=0.2, min=0, max=1.0)
6         ui.button("Run", color="primary", on_click=_run_rag)
```

Please tell me how to add a button to this NiceGUI application.

Ol gpt-oss:120b Today at 3:10 PM

Thought for 1 seconds ▾

Below is a short, step-by-step guide that shows **exactly how** to add another button (or any number of buttons) to the snippet you posted.

It works with the current NiceGUI version ( $\geq 1.4$ ) and follows the same style you already use.

## 1 The “big picture”

```
text
1 ui.row()      # a horizontal container (your current UI line)
2 |
3 |  ui.switch() # ...and the other controls you already have
4 |  ui.number()
5 |  ui.button() # the new button you want to add
```

A button can be placed:

- **inside the same `ui.row()`** – it will appear next to the other controls.
- **in a separate `ui.row()`** (or any other layout container) – this is handy if you want the button on its own line or grouped with other buttons.

Both ways are shown below.

# ADDING DATA TO AUGMENTED GE

- Example:
  - Asking ChatGPT about code:
    1. Python code
    2. Question
  - Response based on context, i.e.,
    - Python code
    - Question

```
python
1 v with ui.row():
2             _switch_augment_question = ui.switch('Augment question')
3             _switch_augment_dataset = ui.switch('Augment context', value=True)
4             _choose_top_k = ui.number("Number of chunks", value=10, min=1, max=100)
5             _choose_min_sim = ui.number("Minimum similarity", value=0.2, min=0, max=1.0)
6             ui.button("Run", color="primary", on_click=_run_rag)
```

Please tell me how to add a button to this NiceGUI application.

01 gpt-oss:120b Today at 3:10 PM

Thought for 1 seconds ▾

Below is a short, step-by-step guide that shows **exactly how** to add another button (or any number of buttons) to the snippet you posted.

It works with the current NiceGUI version ( $\geq 1.4$ ) and follows the same style you already use.

## 1 The “big picture”

```
text
1 ui.row()      # a horizontal container (your current UI line)
2 |
3 |  ui.switch() # ...and the other controls you already have
4 |  ui.number()
5 |  ui.button() # the new button you want to add
```

A button can be placed:

- **inside the same `ui.row()`** – it will appear next to the other controls.
- **in a separate `ui.row()`** (or any other layout container) – this is handy if you want the button on its own line or grouped with other buttons.

Both ways are shown below.

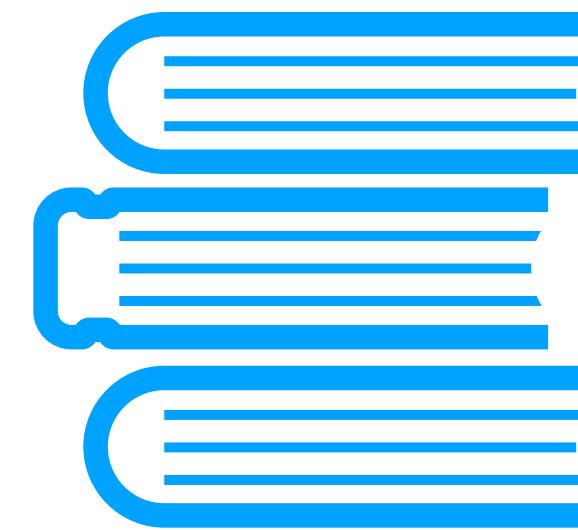
# ADDING DATA TO AN LLM: RETRIEVAL AUGMENTED GENERATION (RAG)

---

# ADDING DATA TO AN LLM: RETRIEVAL AUGMENTED GENERATION (RAG)

---

- Having a dataset of many specific documents



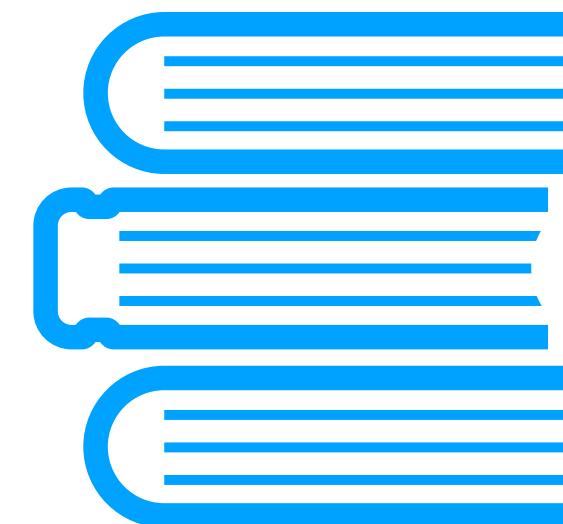
Huge specific dataset

# ADDING DATA TO AN LLM: RETRIEVAL AUGMENTED GENERATION (RAG)

- Having a dataset of many specific documents
- Answering a question does not always require to consider all documents in the dataset



Specific  
document(s)

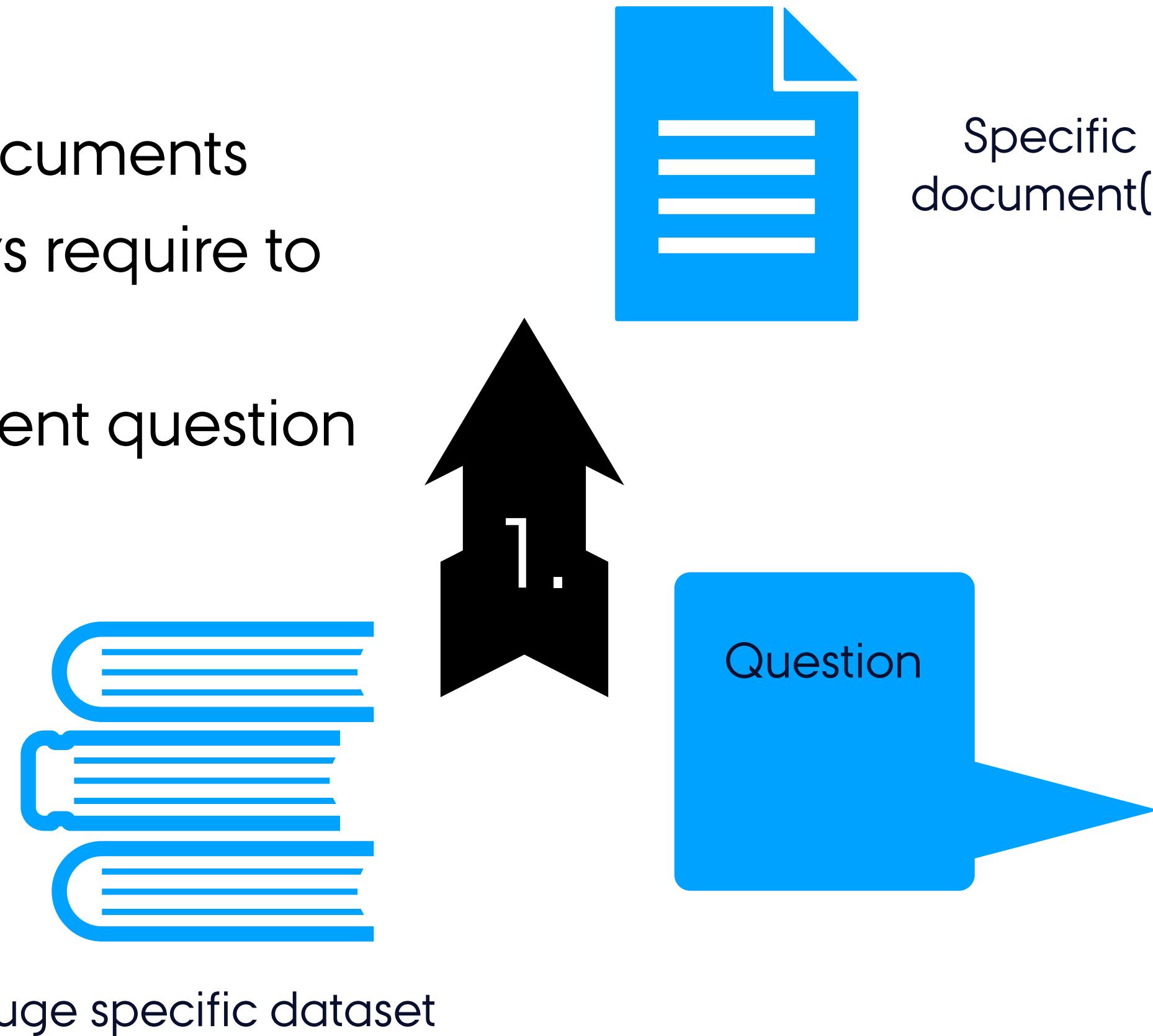


Huge specific dataset



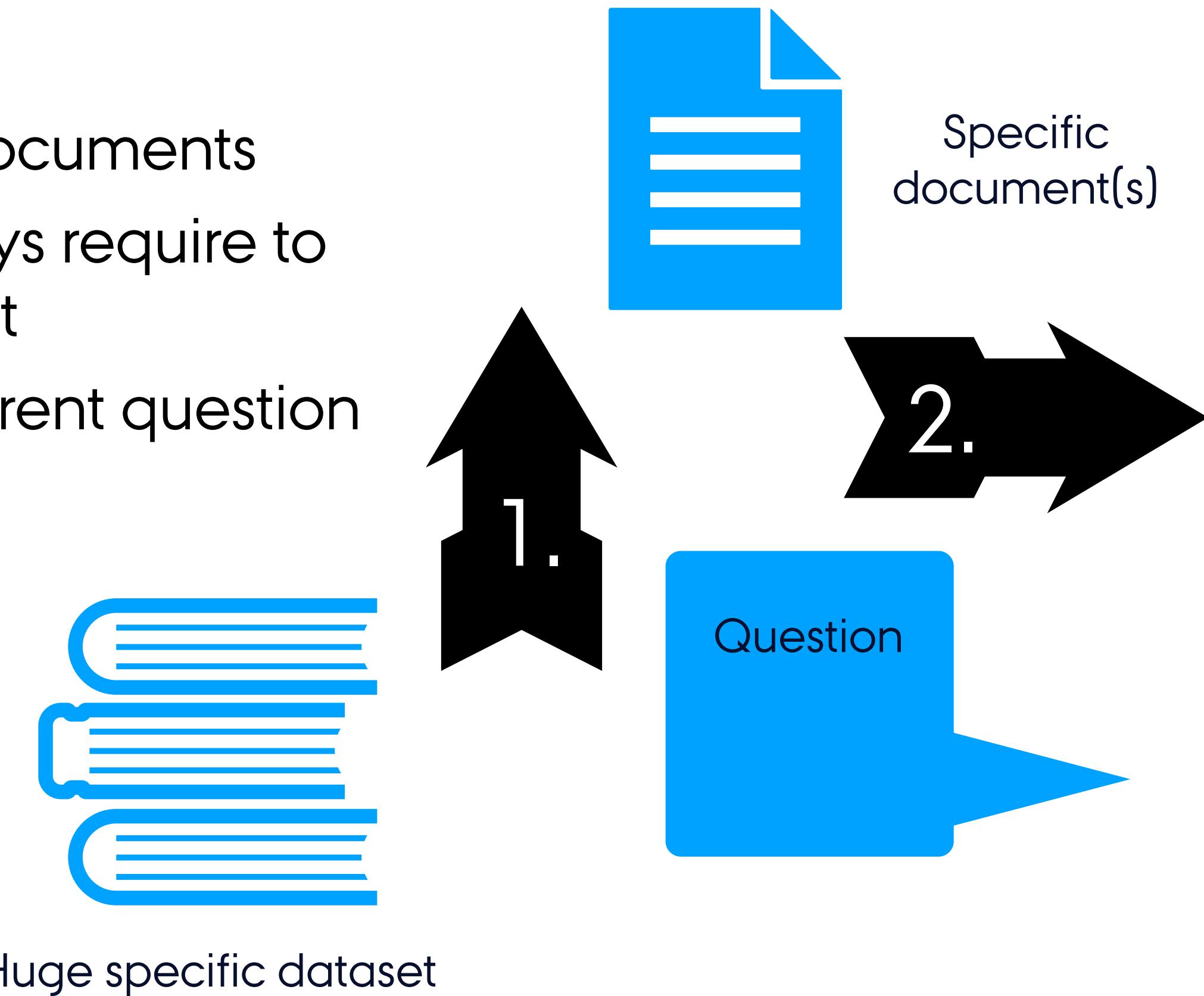
# ADDING DATA TO AN LLM: RETRIEVAL AUGMENTED GENERATION (RAG)

- Having a dataset of many specific documents
- Answering a question does not always require to consider all documents in the dataset
  1. Identify (retrieve) the relevant documents



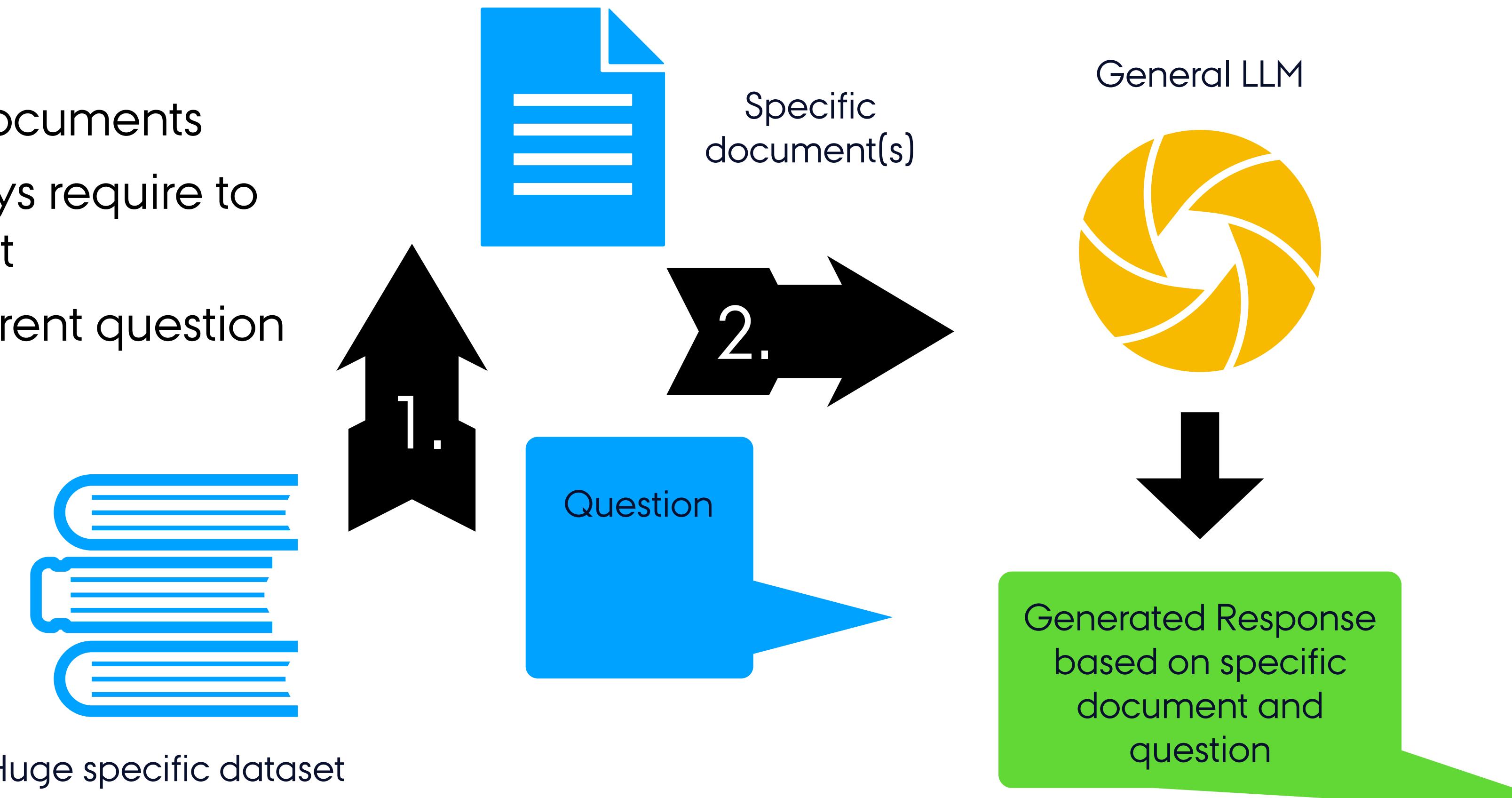
# ADDING DATA TO AN LLM: RETRIEVAL AUGMENTED GENERATION (RAG)

- Having a dataset of many specific documents
- Answering a question does not always require to consider all documents in the dataset
  1. Identify (retrieve) the relevant documents
  2. Send the current relevant documents together with the question to the LLM



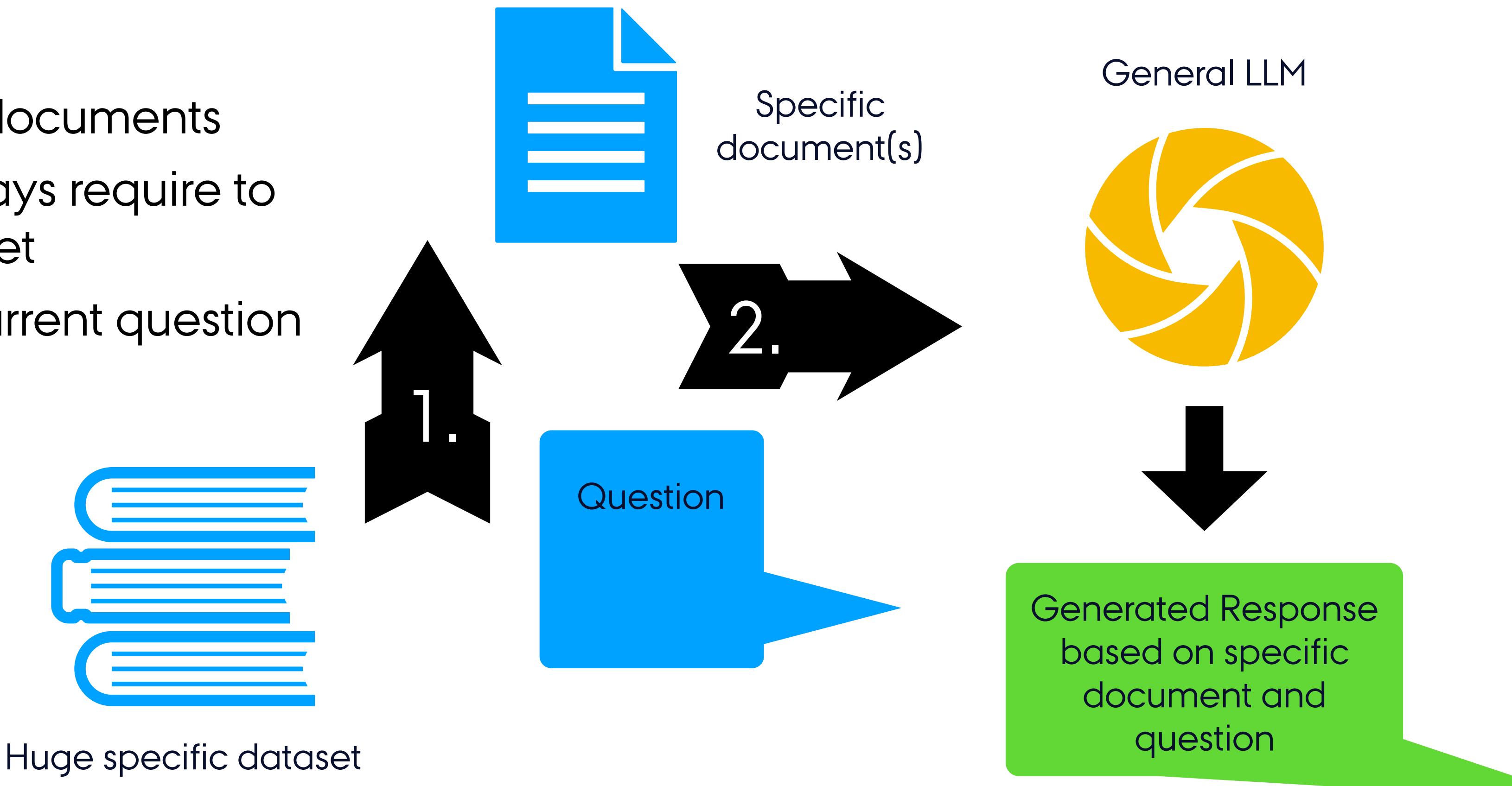
# ADDING DATA TO AN LLM: RETRIEVAL AUGMENTED GENERATION (RAG)

- Having a dataset of many specific documents
- Answering a question does not always require to consider all documents in the dataset
  1. Identify (retrieve) the relevant documents
  2. Send the current relevant documents together with the question to the LLM



# ADDING DATA TO AN LLM: RETRIEVAL AUGMENTED GENERATION (RAG)

- Having a dataset of many specific documents
- Answering a question does not always require to consider all documents in the dataset
  1. Identify (retrieve) the relevant documents
  2. Send the current relevant documents together with the question to the LLM
- Difficulty:
  - Identify relevant documents for a specific question

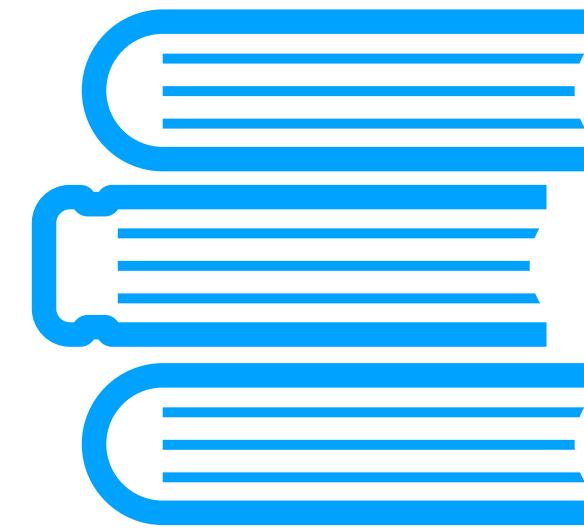


# RAG STEP ONE: INITIALIZE

---

# RAG STEP ONE: INITIALIZE

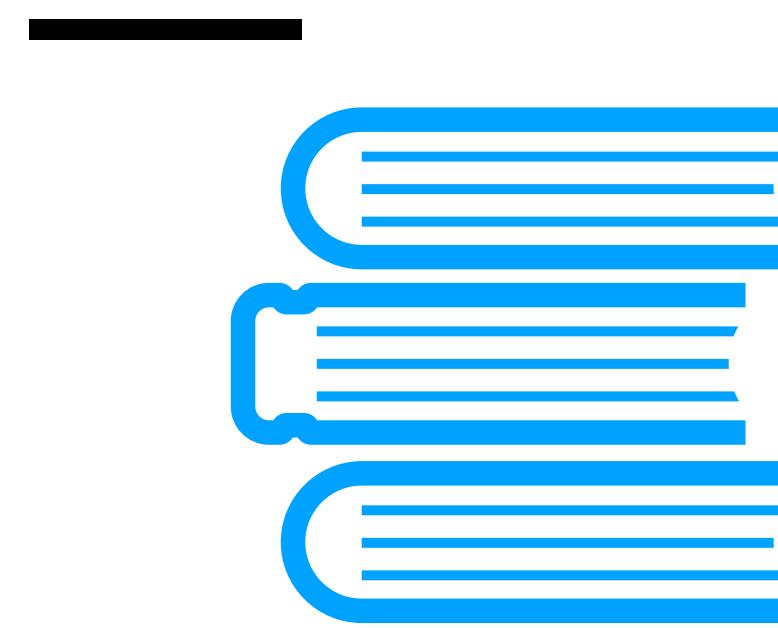
---



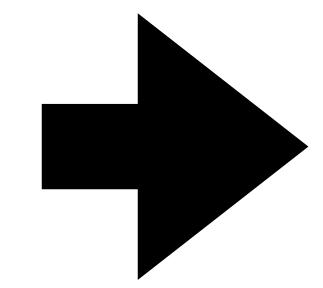
Huge specific dataset

- Initialize RAG for a dataset

# RAG STEP ONE: INITIALIZE



Huge specific dataset



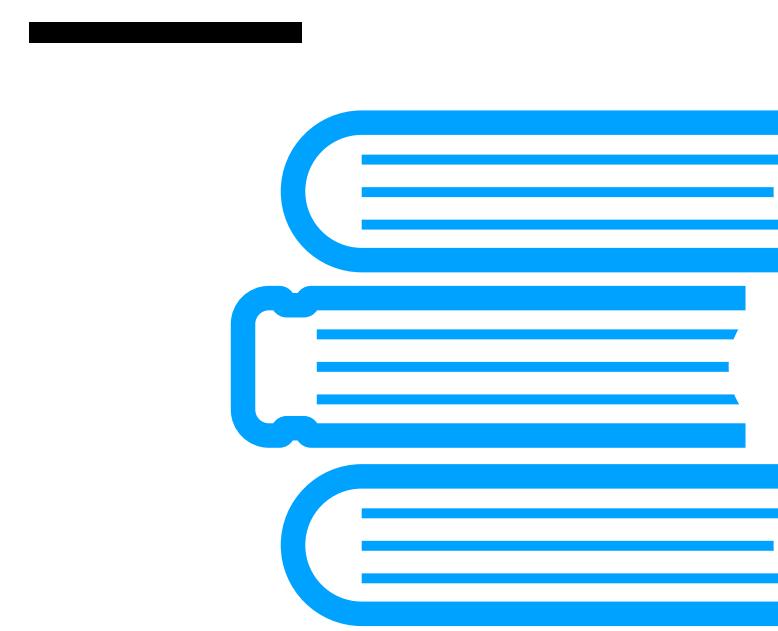
Chunking



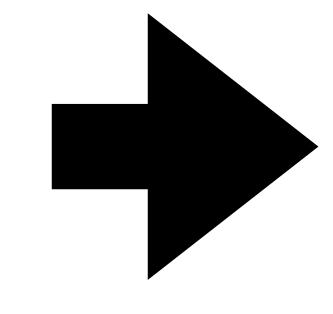
Chunks (parts) of the dataset

- Initialize RAG for a dataset
  - Split the dataset in smaller chunks (size of chunks can be changed, chunks may overlap or be adjacent)

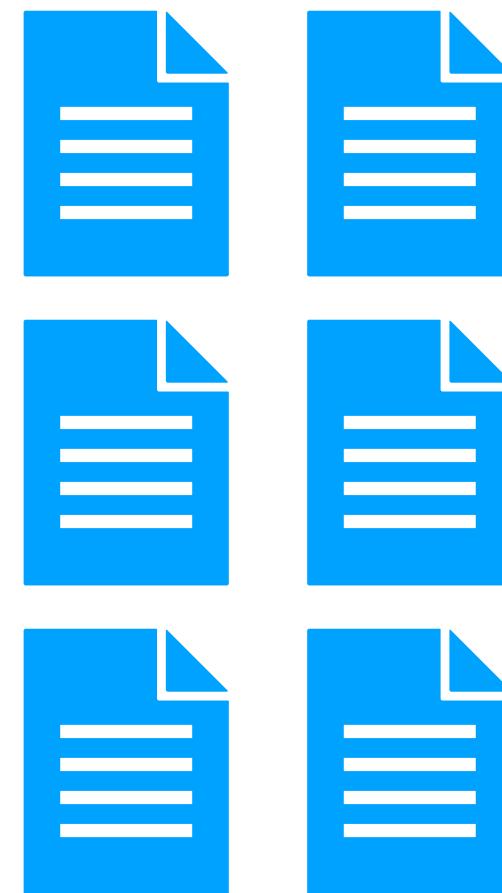
# RAG STEP ONE: INITIALIZE



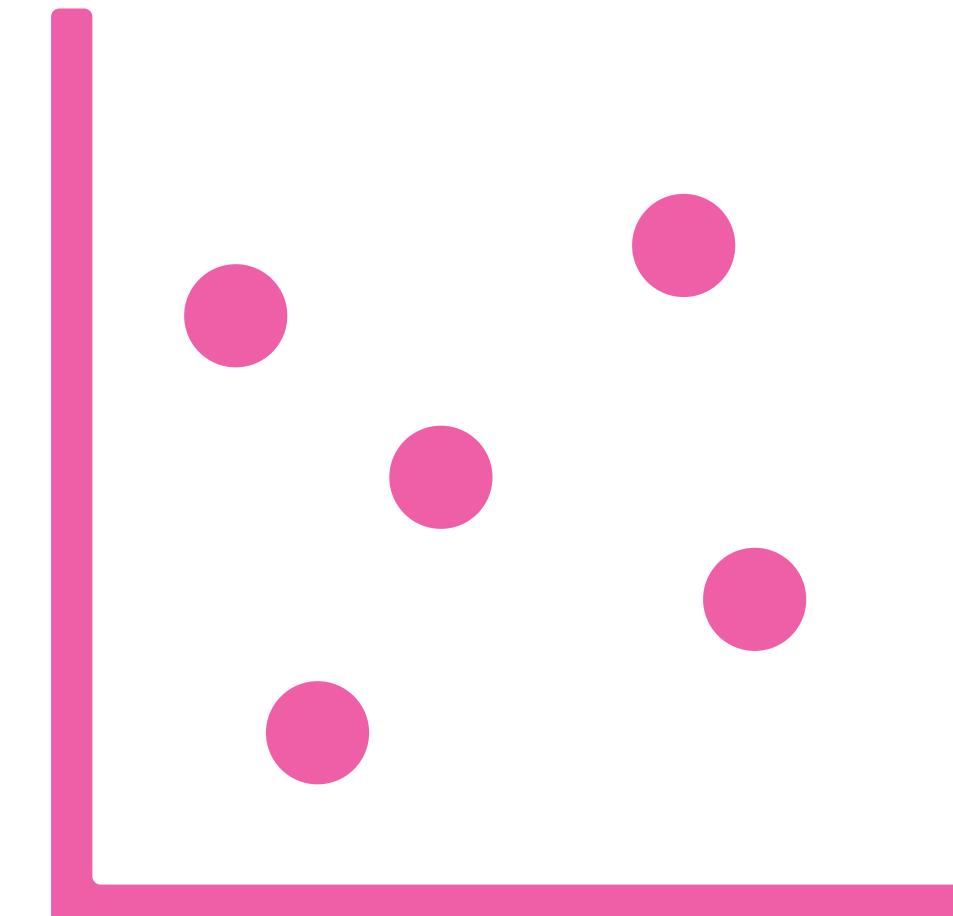
Huge specific dataset



Chunking



Chunks (parts) of the dataset



Embeddings (vector space locations) of the each chunk

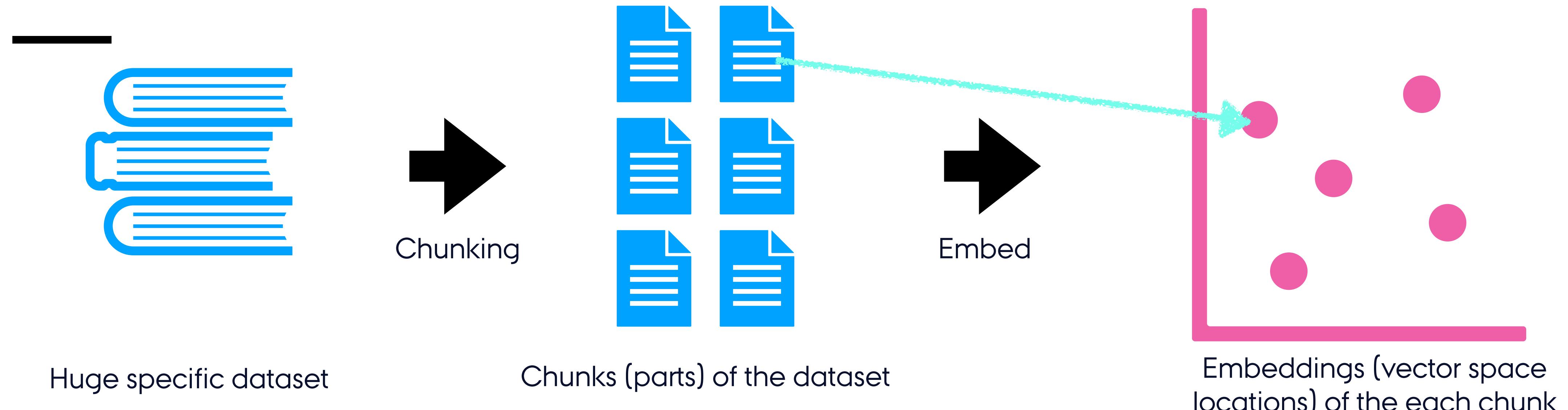
- Initialize RAG for a dataset
  - Split the dataset in smaller chunks (size of chunks can be changed, chunks may overlap or be adjacent)

# RAG STEP ONE: INITIALIZE



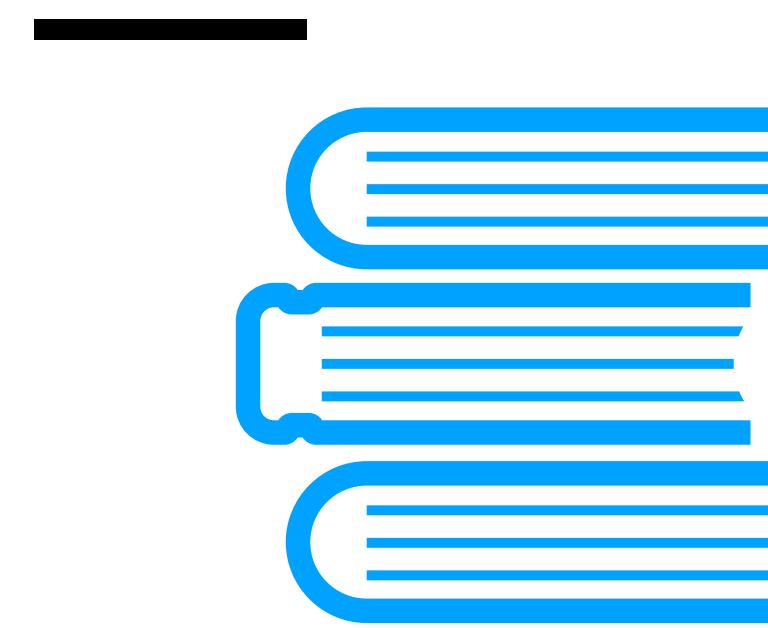
- Initialize RAG for a dataset
  - Split the dataset in smaller chunks (size of chunks can be changed, chunks may overlap or be adjacent)
  - Embed each chunks in a vector space

# RAG STEP ONE: INITIALIZE

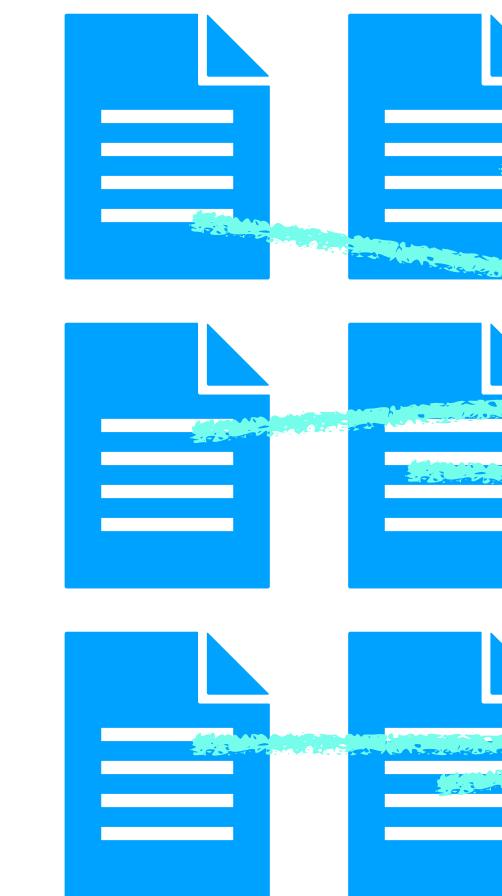


- Initialize RAG for a dataset
  - Split the dataset in smaller chunks (size of chunks can be changed, chunks may overlap or be adjacent)
  - Embed each chunks in a vector space
    - Remember from Lecture 2: Embedding for tokens of LLM inputs
    - Place a sentence/ sequence of words in a vector space, s.t., similar sequence are next to each other

# RAG STEP ONE: INITIALIZE

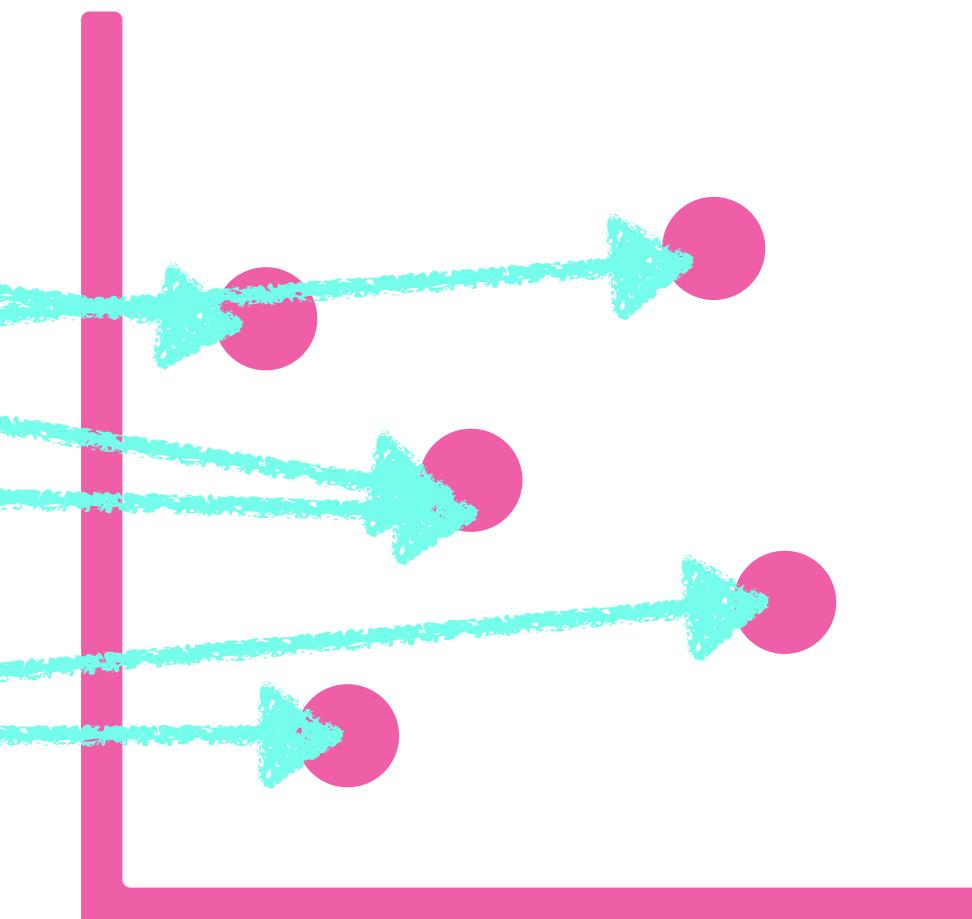


→  
Chunking



Chunks (parts) of the dataset

→  
Embed



Embeddings (vector space locations) of the each chunk

- Initialize RAG for a dataset
  - Split the dataset in smaller chunks (size of chunks can be changed, chunks may overlap or be adjacent)
  - Embed each chunks in a vector space
    - Remember from Lecture 2: Embedding for tokens of LLM inputs
    - Place a sentence/ sequence of words in a vector space, s.t., similar sequence are next to each other

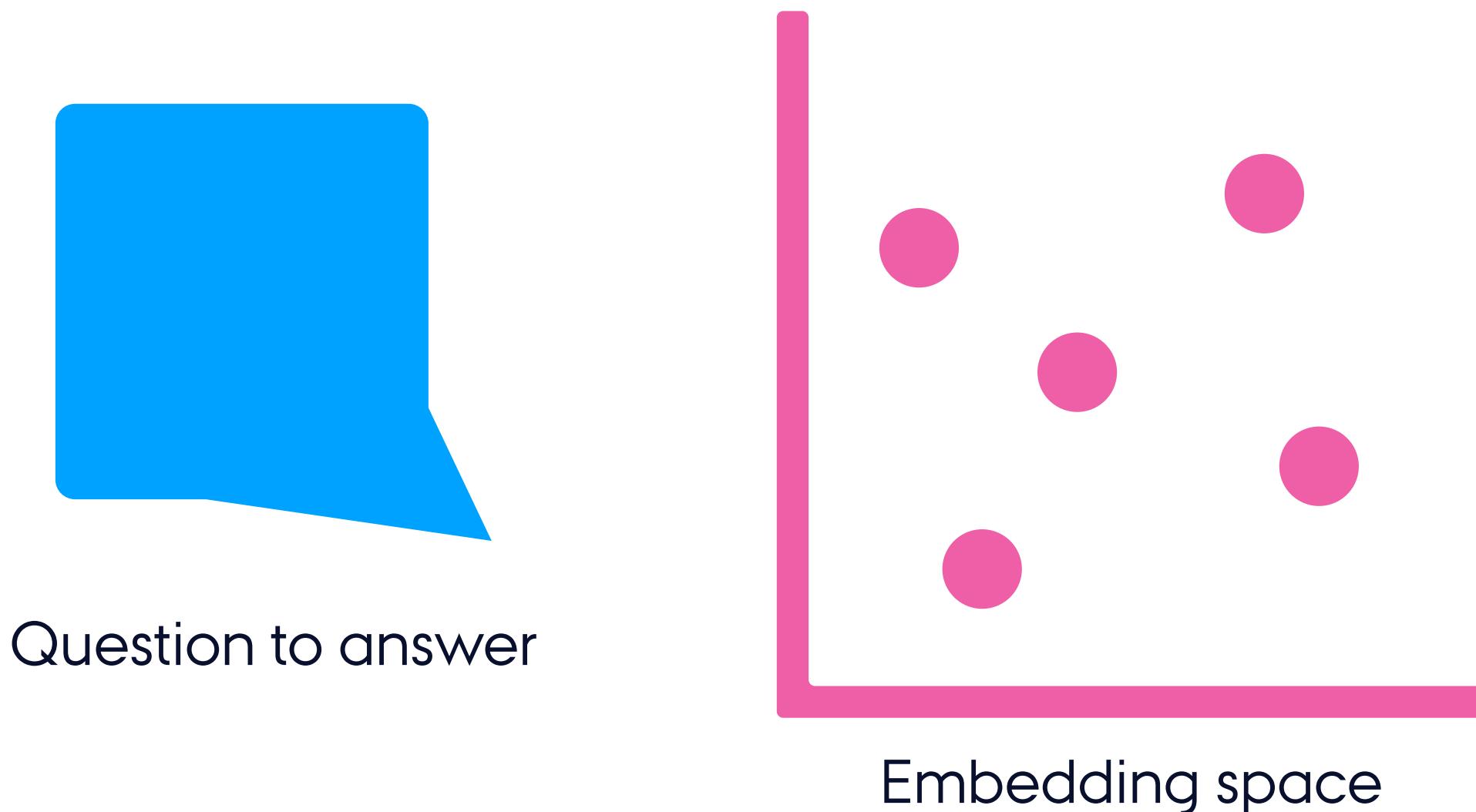
# RAG STEP TWO: RETRIEVE & AUGMENT

---

- Generate a response with augmented context

# RAG STEP TWO: RETRIEVE & AUGMENT

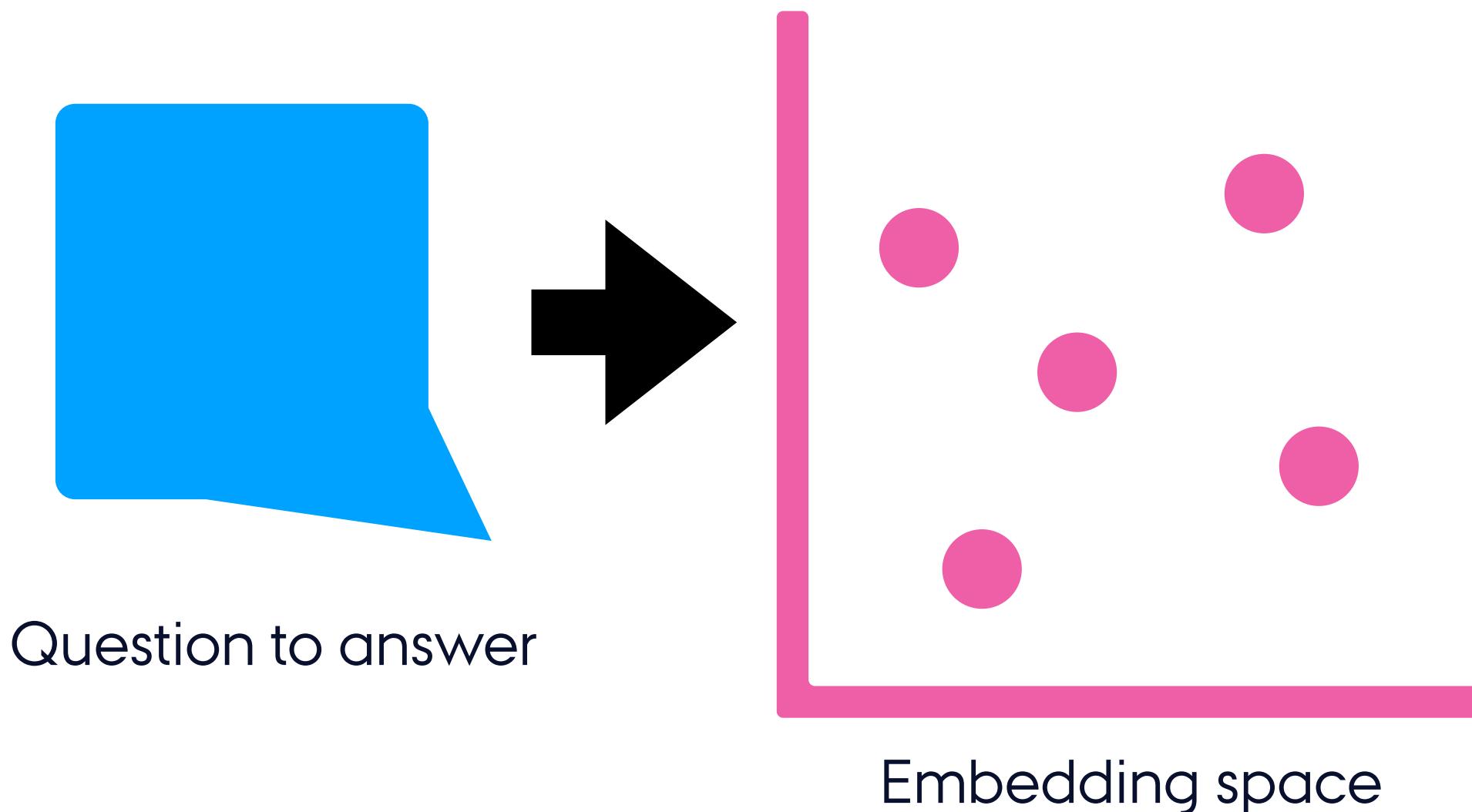
---



- Generate a response with augmented context

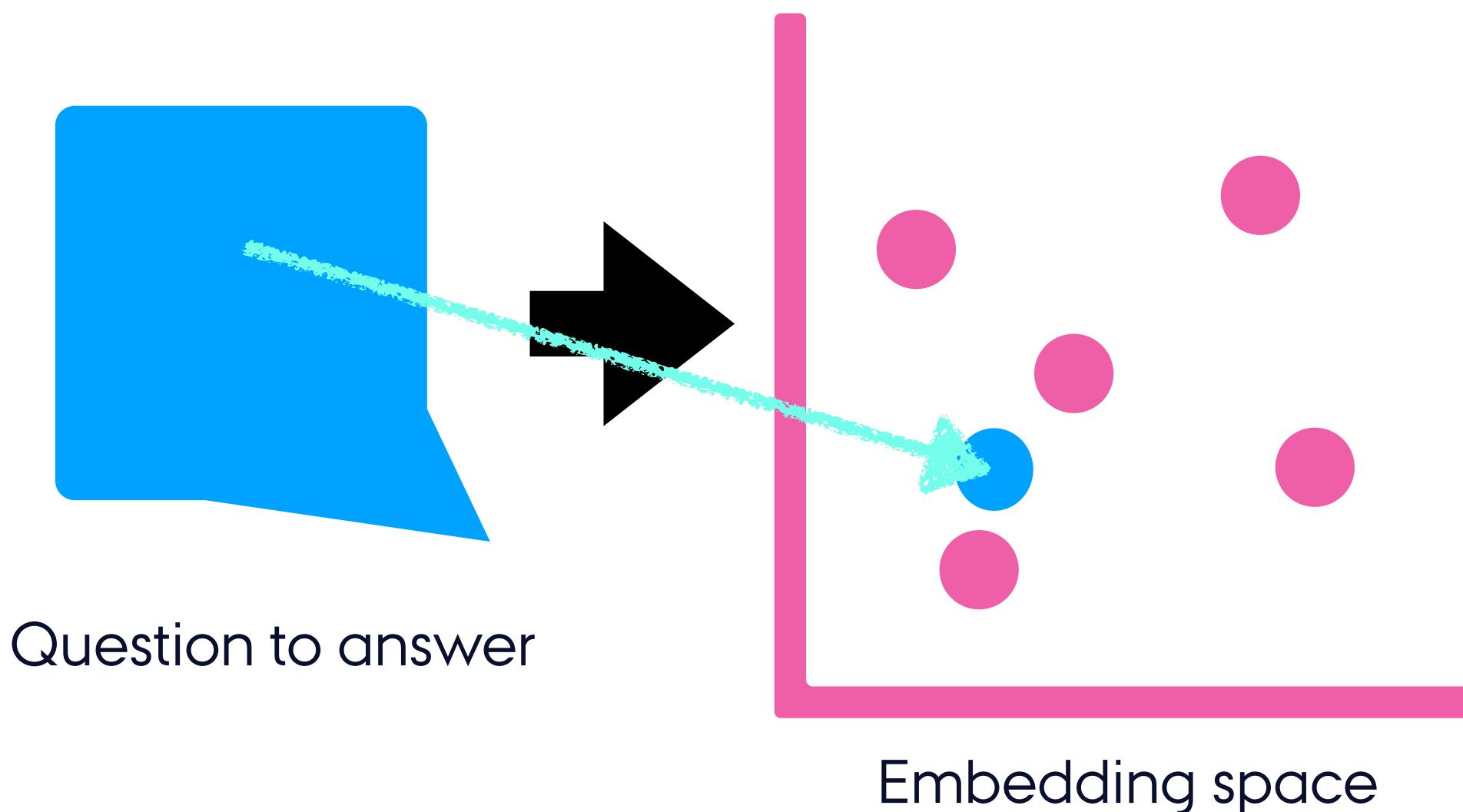
# RAG STEP TWO: RETRIEVE & AUGMENT

---



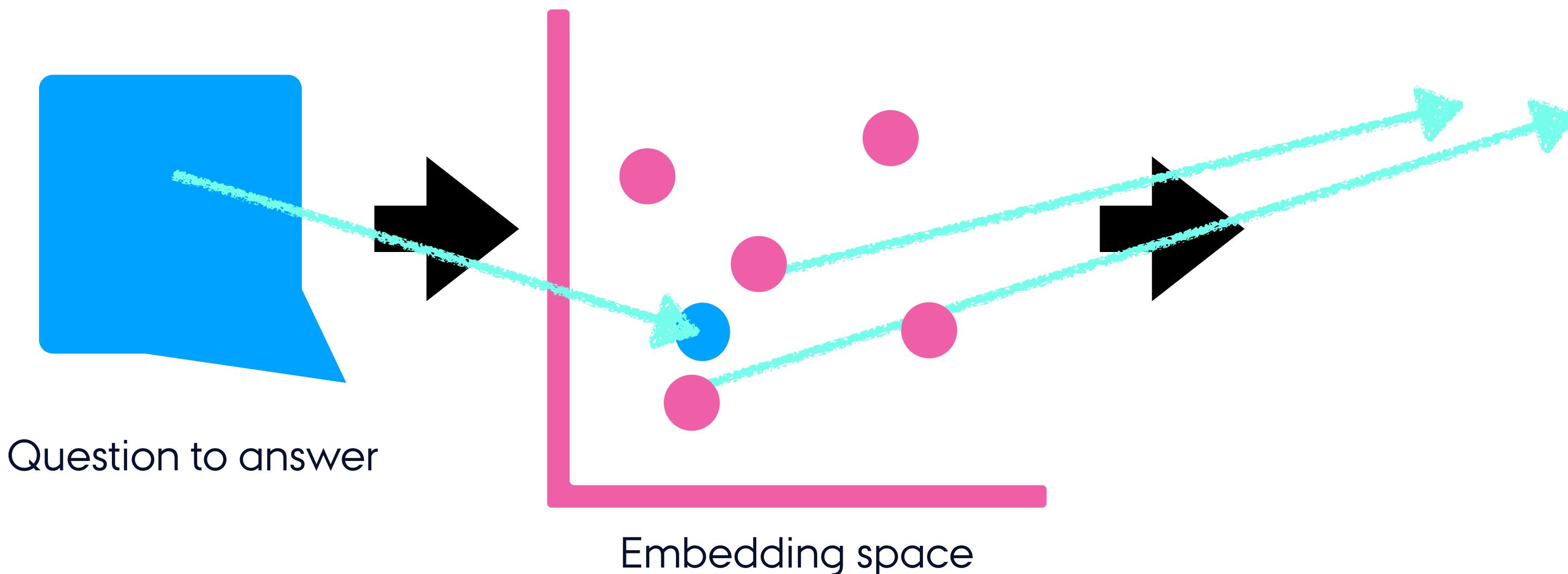
- Generate a response with augmented context
  - Create the embedding of the questions and identify which chunks from dataset are *similar*

# RAG STEP TWO: RETRIEVE & AUGMENT



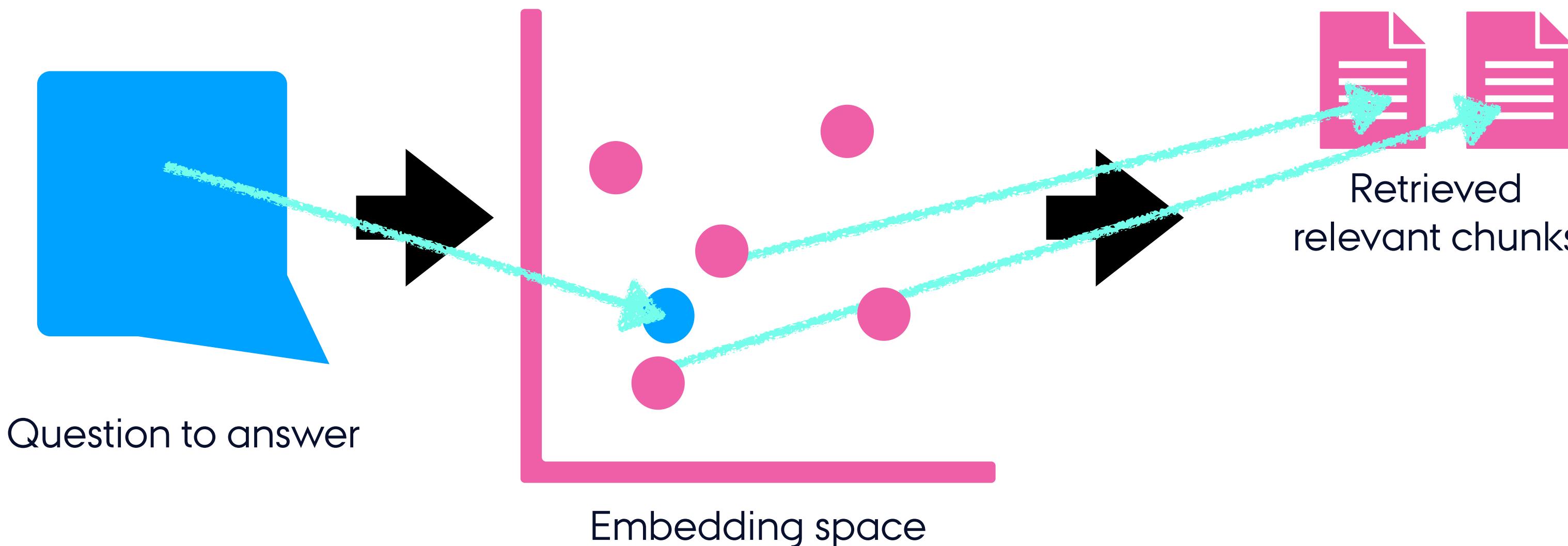
- Generate a response with augmented context
  - Create the embedding of the questions and identify which chunks from dataset are *similar*

# RAG STEP TWO: RETRIEVE & AUGMENT



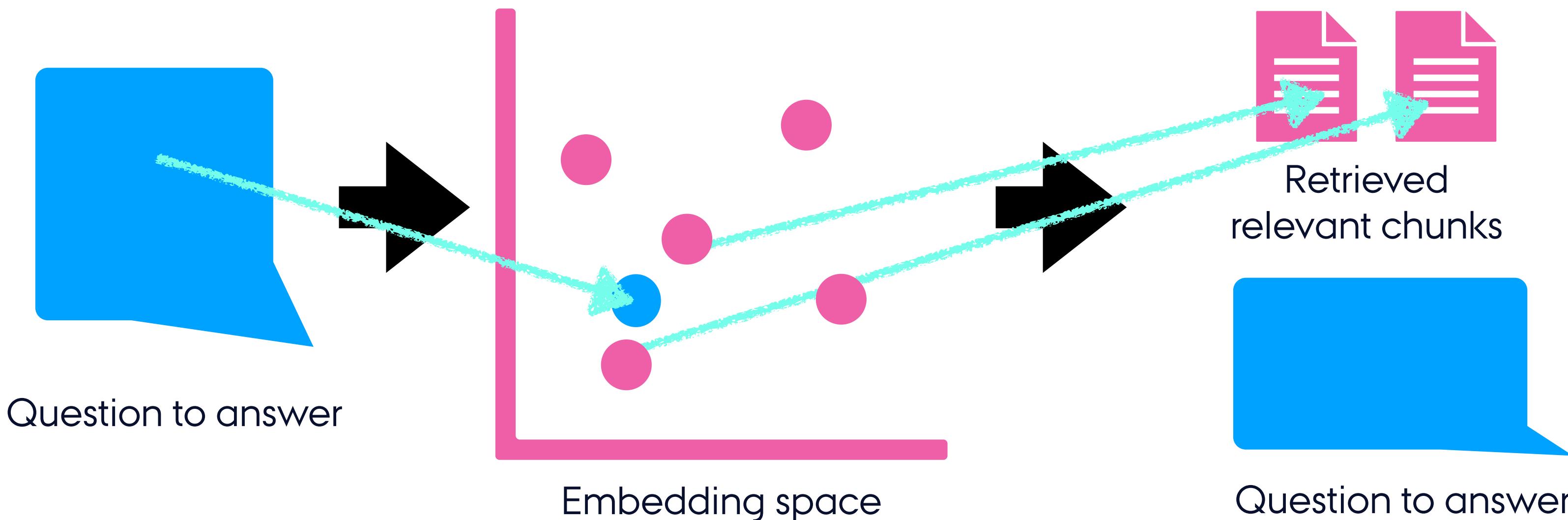
- Generate a response with augmented context
  - Create the embedding of the questions and identify which chunks from dataset are *similar*

# RAG STEP TWO: RETRIEVE & AUGMENT



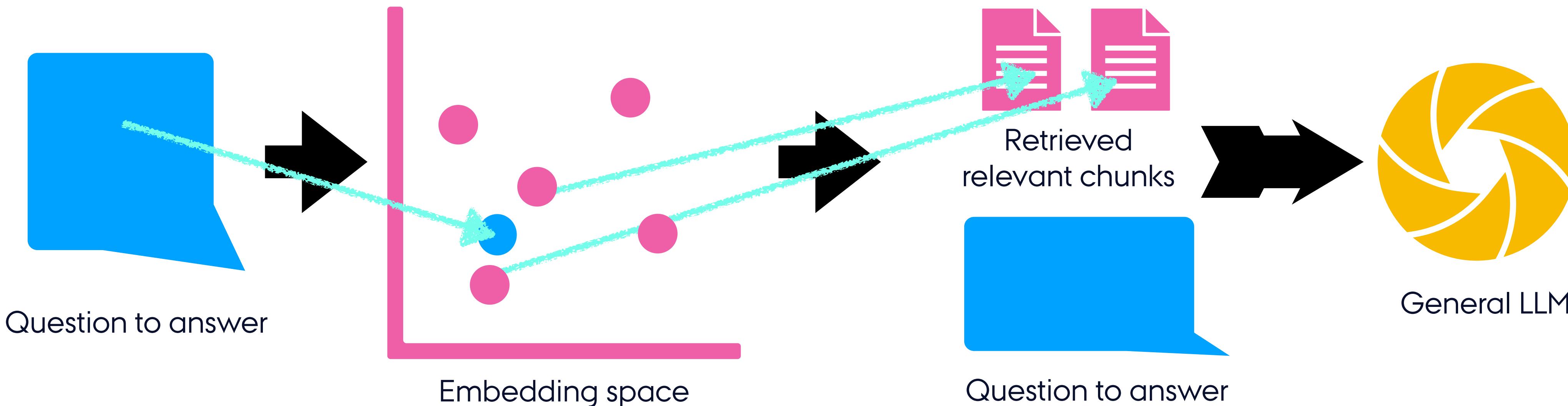
- Generate a response with augmented context
  - Create the embedding of the questions and identify which chunks from dataset are *similar*

# RAG STEP TWO: RETRIEVE & AUGMENT



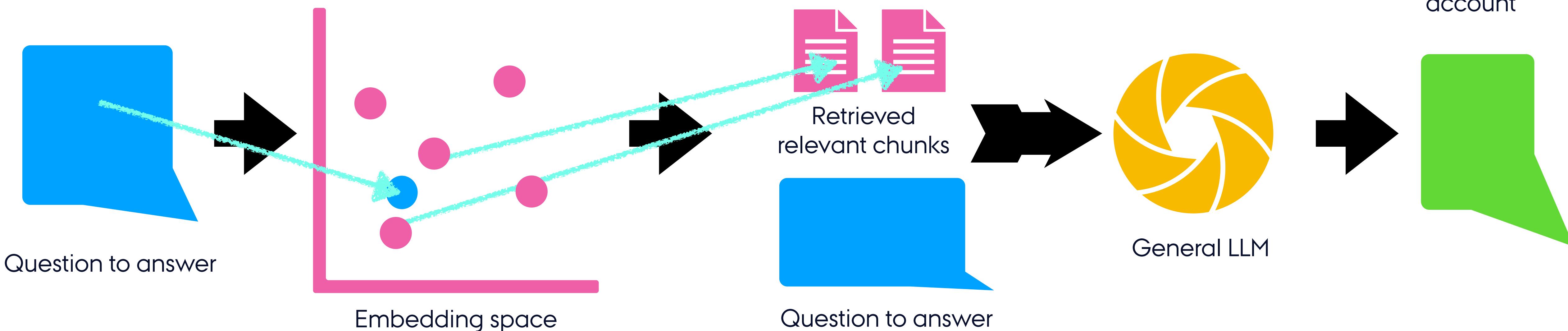
- Generate a response with augmented context
  - Create the embedding of the questions and identify which chunks from dataset are *similar*
  - Augment the context for the LLM with these chunks

# RAG STEP TWO: RETRIEVE & AUGMENT



- Generate a response with augmented context
  - Create the embedding of the questions and identify which chunks from dataset are *similar*
  - Augment the context for the LLM with these chunks

# RAG STEP TWO: RETRIEVE & AUGMENT



# RAG IN PRACTICE

---

- Let's try it together

# RAG EXAMPLE APP ON UCLOUD

Small RAG Example

Define the data to search via RAG

Add data sources: 45.7KB / 100.00%

AU-Encyclopedia-small.pdf 45.7KB / 100.00%

Chunk size (characters): 250

Overlapping chunks

Data sources currently in use

**!** No sources

Small RAG app using the OpenAI API.

# RAG EXAMPLE APP ON UCLOUD

Small RAG Example

Define the data to search via RAG

Add data sources: 45.7KB / 100.00%

AU-Encyclopedia-small.pdf 45.7KB / 100.00%

Chunk size (characters): 250

Overlapping chunks

Data sources currently in use

No sources

Small RAG app using the OpenAI API.

The app supports multiple PDF and plain text files as dataset for RAG.

# RAG EXAMPLE APP ON UCLOUD

Small RAG Example

Define the data to search via RAG

Add data sources: 45.7KB / 100.00%

AU-Encyclopedia-small.pdf 45.7KB / 100.00%

Chunk size (characters): 250

Overlapping chunks

Data sources currently in use

No sources

Small RAG app using the OpenAI API.

The size of each chunk: The text in the dataset will be split by parts of this size.

# RAG EXAMPLE APP ON UCLOUD

Small RAG Example

Define the data to search via RAG

Add data sources: 45.7KB / 100.00%

AU-Encyclopedia-small.pdf 45.7KB / 100.00%

Chunk size (characters): 250

Overlapping chunks

Data sources currently in use

No sources

Small RAG app using the OpenAI API.

Should the chunks be adjacent or overlap each other?

# RAG EXAMPLE APP ON UCLOUD

Small RAG Example

Define the data to search via RAG

Add data sources: 45.7KB / 100.00%

AU-Encyclopedia-small.pdf 45.7KB / 100.00%

Chunk size (characters): 250

Overlapping chunks

Data sources currently in use

**!** No sources

Small RAG app using the OpenAI API.

# RAG EXAMPLE APP ON UCLOUD

Small RAG Example

Define the data to search via RAG

Add data sources: 0.0B / 0.00%

Chunk size (characters):

Overlapping chunks

RESET RAG DATABASE INITIALIZE RAG DATABASE

Data sources currently in use

AU-Encyclopedia-small.pdf

127.0.0.1

RAG-based chat

Ask a question:  
What is Aarhus BSS?

Augment question Augment context Number of chunks 10 Minimum similarity 0.2 RUN

Results

# RAG EXAMPLE APP ON UCLOUD

Small RAG Example

Define the data to search via RAG

Add data sources: 0.0B / 0.00%

Chunk size (characters):

Overlapping chunks

**RESET RAG DATABASE** **INITIALIZE RAG DATABASE**

Data sources currently in use

AU-Encyclopedia-small.pdf

RAG-based chat

Ask a question:  
What is Aarhus BSS?

Augment question  Augment context

Number of chunks: 10 Minimum similarity: 0.2

**RUN**

Use ChatGPT to  
rephrase the  
question.

# RAG EXAMPLE APP ON UCLOUD

Small RAG Example

Define the data to search via RAG

Add data sources: 0.0B / 0.00%

Chunk size (characters):

Overlapping chunks

RESET RAG DATABASE INITIALIZE RAG DATABASE

Data sources currently in use

AU-Encyclopedia-small.pdf

RAG-based chat

Ask a question: What is Aarhus BSS?

Augment question  Augment context  Number of chunks 10  Minimum similarity 0.2

Results

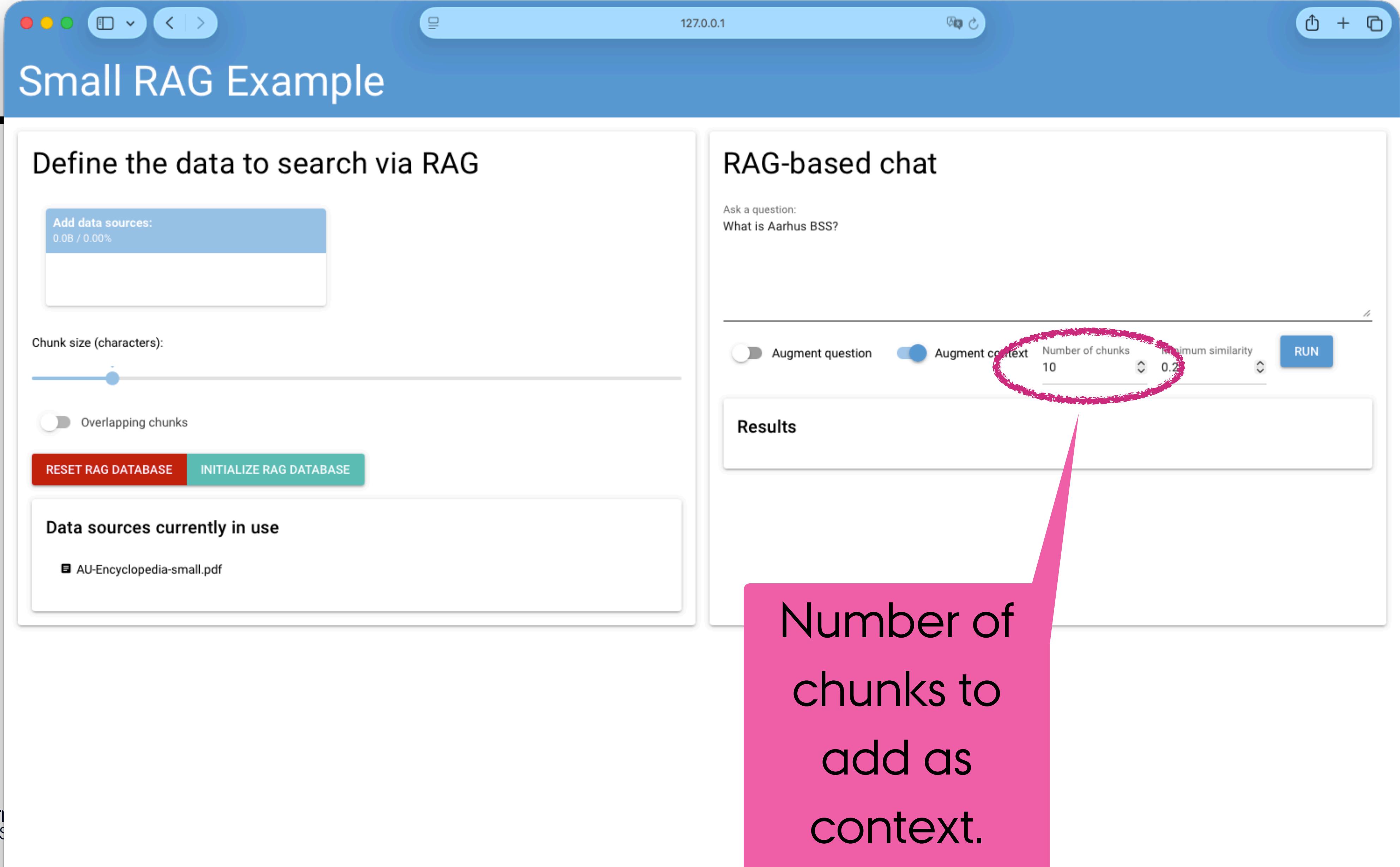
Use the RAG augmentation.

AARHUS BSS DEPARTMENT OF BUSINESS AND ECONOMICS

ASSOCIATION OF MBAs ACCREDITED

EFMD EQUIS ACCREDITED

# RAG EXAMPLE APP ON UCLOUD



Small RAG Example

Define the data to search via RAG

Add data sources: 0.0B / 0.00%

Chunk size (characters):

Overlapping chunks

**RESET RAG DATABASE** **INITIALIZE RAG DATABASE**

Data sources currently in use

AU-Encyclopedia-small.pdf

RAG-based chat

Ask a question:  
What is Aarhus BSS?

Augment question  Augment context **Number of chunks** 10  Maximum similarity 0.2 **RUN**

Results

Number of chunks to add as context.

AARHUS BSS DEPARTMENT OF AARHUS

ASSOCIATION OF AMBAs ACCREDITED

EFMD EQUIS ACCREDITED

# RAG EXAMPLE APP ON UCLOUD

The screenshot shows a web application interface for a RAG (Retrieval-Augmented Generation) example. The left section, titled 'Define the data to search via RAG', includes a 'Add data sources:' button, a progress bar showing '0.0B / 0.00%', a 'Chunk size (characters)' slider, an 'Overlapping chunks' toggle, and buttons for 'RESET RAG DATABASE' and 'INITIALIZE RAG DATABASE'. The right section, titled 'RAG-based chat', shows a question 'What is Aarhus BSS?' and a configuration panel with 'Augment question' and 'Augment context' toggles, a 'Number of chunks' input set to 10, a 'Minimum similarity' input set to 0.2 (circled in red), and a 'RUN' button. A pink callout box points to the 'Minimum similarity' input field.

Minimal required similarity between question and chunk to add chunk to context.

# EXAMPLE APP ON UCLOUD: TRY IT

---

- I will give a short demonstration
- Play around with it on your own!
  - You will find it in `Tutorials/Lecture_9`
  - Run the `main.py` to open NiceGUI application
    - ▶ It will ask for your OpenAI API key
  - There is a `pdfs.zip` available (Right click → Download → Extract on your PC)
    - ▶ You may use your own PDFs (remember, everything is sent to *uCloud* and *OpenAI*!)
    - ▶ Use smaller PDFs, i.e., not too many pages and too much text
    - ▶ Large PDFs, especially together with small chunk sizes, will take a lot of time to initialize RAG
- You may take a look at the source code of the app, but it is rather complex → We will implement a more simple RAG in the next tutorial

# EXAMPLE APP ON UCLOUD: GET A FEELING FOR RAG

---

# EXAMPLE APP ON UCLOUD: GET A FEELING FOR RAG

---

- RAG initialization:
  - What is a good *chunk size* for a PDF containing multiple big paragraphs of the same topic?
  - What if there are many short paragraphs of different topics?
  - When do we need *overlapping chunks*?

# EXAMPLE APP ON UCLOUD: GET A FEELING FOR RAG

---

- RAG initialization:
  - What is a good *chunk size* for a PDF containing multiple big paragraphs of the same topic?
  - What if there are many short paragraphs of different topics?
  - When do we need *overlapping chunks*?
- RAG chat:
  - Why may it be helpful to *augment the question*?
  - What happens without *context augmentation*?
  - What are good values for *number of chunks* and *minimum similarity*?
  - How to identify them?

# RAG EXAMPLE APP ON UCLOUD:

## DEMO 1

Small RAG Example

Define the data to search via RAG

Add data sources: 45.7KB / 100.00%

AU-Encyclopedia-small.pdf 45.7KB / 100.00%

Chunk size (characters):

Overlapping chunks

RESET RAG DATABASE INITIALIZE RAG DATABASE

Data sources currently in use

AU-Encyclopedia-small.pdf

RAG-based chat

Ask a question: What is Aarhus BSS?

Augment question  Augment context  Number of chunks 2 Minimum similarity 0.2 **RUN**

Results

User: What is Aarhus BSS?

GPT: Aarhus BSS is the brand name of the Faculty of Business and Social Sciences at Aarhus University in Denmark. It was formed in 2012 from the merger of the former Aarhus School of Business and the social sciences division and serves as the umbrella for the university's programs and research in business, economics, management, law, and related social sciences (offered at bachelor, master, and PhD levels, in Danish and English).

# RAG EXAMPLE APP ON UCLOUD:

## DEMO 2

The screenshot shows a web-based application for a RAG (Retrieval-Augmented Generation) example. The interface is divided into two main sections: 'Define the data to search via RAG' on the left and 'RAG-based chat' on the right.

**Define the data to search via RAG:**

- Add data sources:** A progress bar shows 45.7KB / 100.00% for 'AU-Encyclopedia-small.pdf'.
- Chunk size (characters):** A slider is set to a low value.
- Overlapping chunks:** A toggle switch is off.
- Buttons:** 'RESET RAG DATABASE' (red) and 'INITIALIZE RAG DATABASE' (green).
- Data sources currently in use:** A list showing 'AU-Encyclopedia-small.pdf'.

**RAG-based chat:**

- Ask a question:** 'What is Aarhus BSS?'
- Configuration:** 'Augment question' is off, 'Augment context' is on, 'Number of chunks' is 2, and 'Minimum similarity' is 0.2.
- Results:**
  - RAG context, 1:** 'Aarhus BSS (School of Business and Social Sciences), Faculty Faculty: There are five faculties at AU. Faculty of Arts, Faculty of F'
  - RAG context, 2:** 'n-making process that's always completely rational and reasonable, of course... Eduroam: AU's Wi-Fi network. Good luck! Emdrup: One of AU's campuses, located just outside Copenhagen. In other words, a tiny AU oasis in the middle of a University'
  - GPT:** 'Aarhus BSS is the School of Business and Social Sciences.'

# RAG EXAMPLE APP ON UCLOUD:

## DEMO 3

The screenshot shows the 'Small RAG Example' application interface. The left section, titled 'Define the data to search via RAG', allows users to add data sources (currently 'AU-Encyclopedia-small.pdf' at 45.7KB / 100.00%), set chunk size (512 characters), and choose overlapping chunks. It includes 'RESET RAG DATABASE' and 'INITIALIZE RAG DATABASE' buttons. The right section, titled 'RAG-based chat', shows a user input 'ECTS' and a 'RUN' button. Below, the 'Results' section displays three responses: 'RAG context, 1' (a definition of ECTS), 'RAG context, 2' (a statement about ECTS being 30 credits), and 'GPT' (a definition of ECTS as 180 credits over six semesters). A 'User ECTS' label is also present.

Small RAG Example

Define the data to search via RAG

Add data sources: 45.7KB / 100.00%

AU-Encyclopedia-small.pdf 45.7KB / 100.00%

Chunk size (characters): 512

Overlapping chunks

RESET RAG DATABASE INITIALIZE RAG DATABASE

Data sources currently in use

AU-Encyclopedia-small.pdf

RAG-based chat

Ask a question: ECTS

Augment question Augment context Number of chunks 2 Minimum similarity 0.5 RUN

Results

User ECTS

RAG context, 1

at place to work. E ECTS: ETCS, ESCT, ETSC or just E-C-T-S. Can you count to 180 ECTS? ECTS stands for the European Credit Transfer and and Accumulation System. It's a standard system for comparing academic credits (which measure the workload f

RAG context, 2

or a particular course) for higher education throughout the EU. A semester's worth of academic work is equal to 30 ECTS. This means your lecturers decide how much you need to study (and learn) in a single semester based on this number. A decisio

GPT

ECTS stands for the European Credit Transfer and Accumulation System. It's a standard for comparing academic credits in higher education across the EU. A semester's worth of work is 30 ECTS, so 180 ECTS would be six semesters.

# RAG EXAMPLE APP ON UCLOUD:

## DEMO 4

The screenshot shows a web application interface for a RAG (Retrieval-Augmented Generation) example. The top navigation bar includes a back/forward button, a refresh button, and a search bar with the text '127.0.0.1'. The main title 'Small RAG Example' is displayed in a large blue header. The interface is divided into two main sections: 'Define the data to search via RAG' on the left and 'RAG-based chat' on the right.

**Define the data to search via RAG:**

- Add data sources:** A progress bar shows '45.7KB / 100.00%' for 'AU-Encyclopedia-small.pdf'.
- Chunk size (characters):** A slider is set to a low value.
- Overlapping chunks:** A toggle switch is turned off.
- Buttons:** 'RESET RAG DATABASE' (red) and 'INITIALIZE RAG DATABASE' (green).
- Data sources currently in use:** A list showing 'AU-Encyclopedia-small.pdf'.

**RAG-based chat:**

- Ask a question:** The user asks 'What is Health?'
- Configuration:** 'Augment question' is off, 'Augment context' is on, 'Number of chunks' is set to 4, and 'Minimum similarity' is set to 0.2. A 'RUN' button is present.
- Results:** The interface displays three sources of information:
  - RAG context, 1:** 'te term for 'fail''. A failed exam will be registered as Department of Public Health. of departments like the Department of Clinical Medicine and the example, Health is the umbrella for health sciences, and is composed Departments/Schools) special'
  - RAG context, 2:** 'ising in the various disciplines. For or medicine, and is composed of departments (or schools, see umbrella for a variety of related disciplines, such as the humanities of Natural Sciences and Faculty of Technical Sciences. A faculty is an Health'
  - GPT:** 'Health is the umbrella for health sciences.'

# RAG EXAMPLE APP ON UCLOUD:

## DEMO 5

Small RAG Example

Define the data to search via RAG

Add data sources: 45.7KB / 100.00%

AU-Encyclopedia-small.pdf 45.7KB / 100.00%

Chunk size (characters):

Overlapping chunks

RESET RAG DATABASE INITIALIZE RAG DATABASE

Data sources currently in use

AU-Encyclopedia-small.pdf

RAG-based chat

Ask a question:  
What is Health?

Augment question  Augment context  Number of chunks 4 Minimum similarity 0.2 **RUN**

Results

User  
What is Health?

GPT

Health is a state of complete physical, mental, and social well-being, and not merely the absence of disease or infirmity. This is the classic definition from the World Health Organization (1946).

Modern views also describe health as:

- The ability to adapt and manage everyday life and its challenges.
- A dynamic resource for living, influenced by physical, mental, social, environmental, and sometimes spiritual factors.

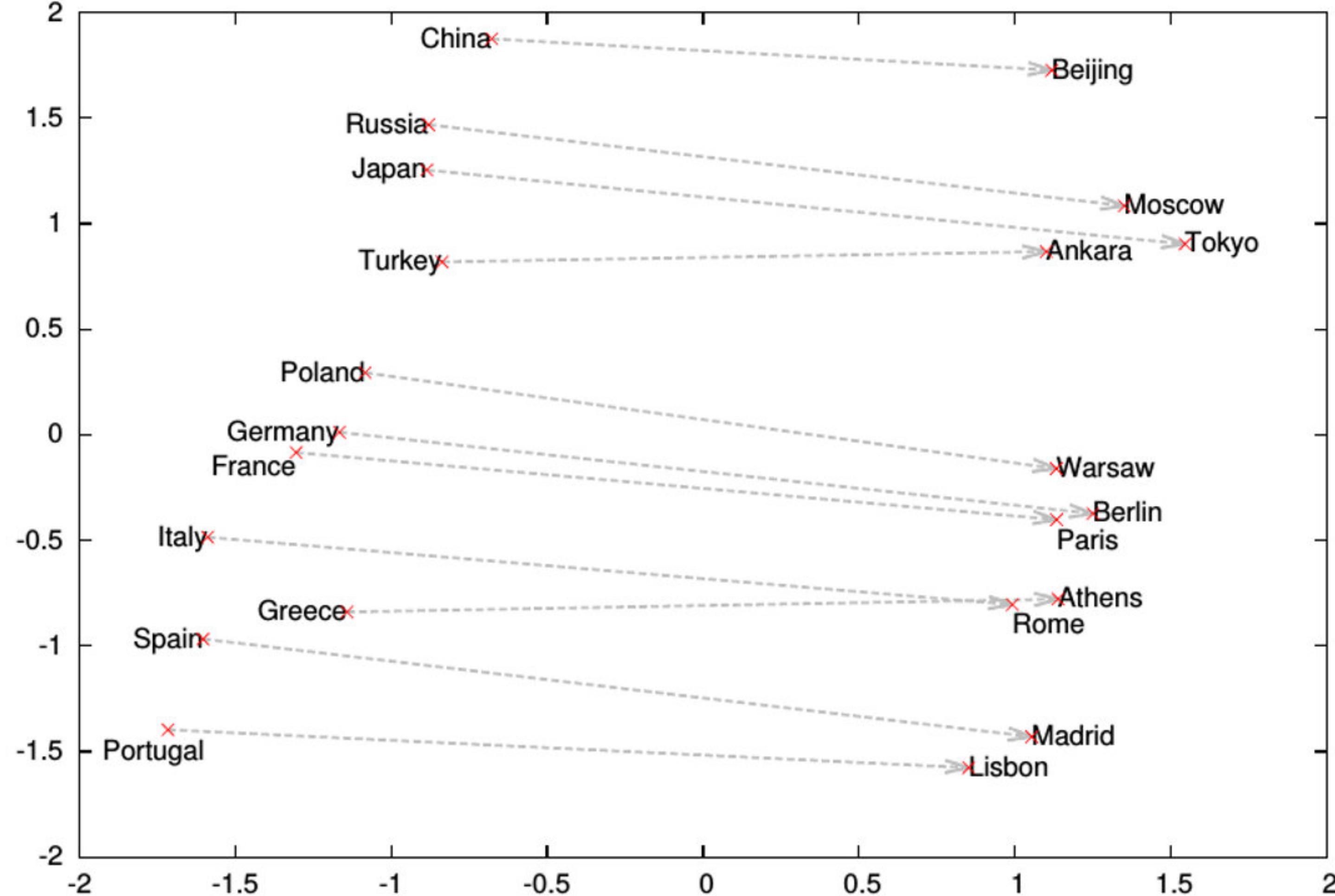
# RAG IN DETAIL

---

- What happens in the *inside*

# RECAP LECTURE 2

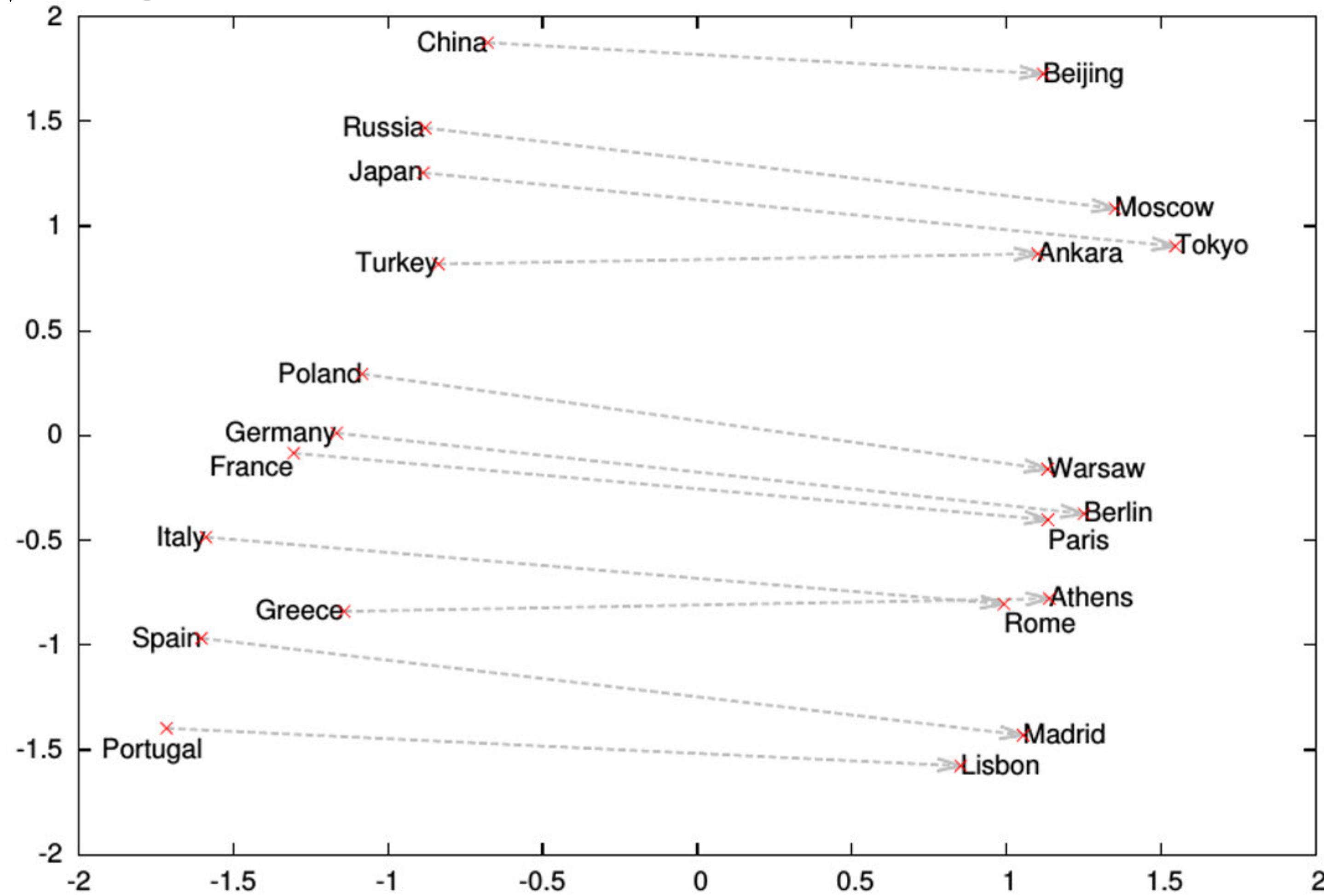
## EXAMPLE: WORD2VEC



# RECAP LECTURE 2

## EXAMPLE: WORD2VEC

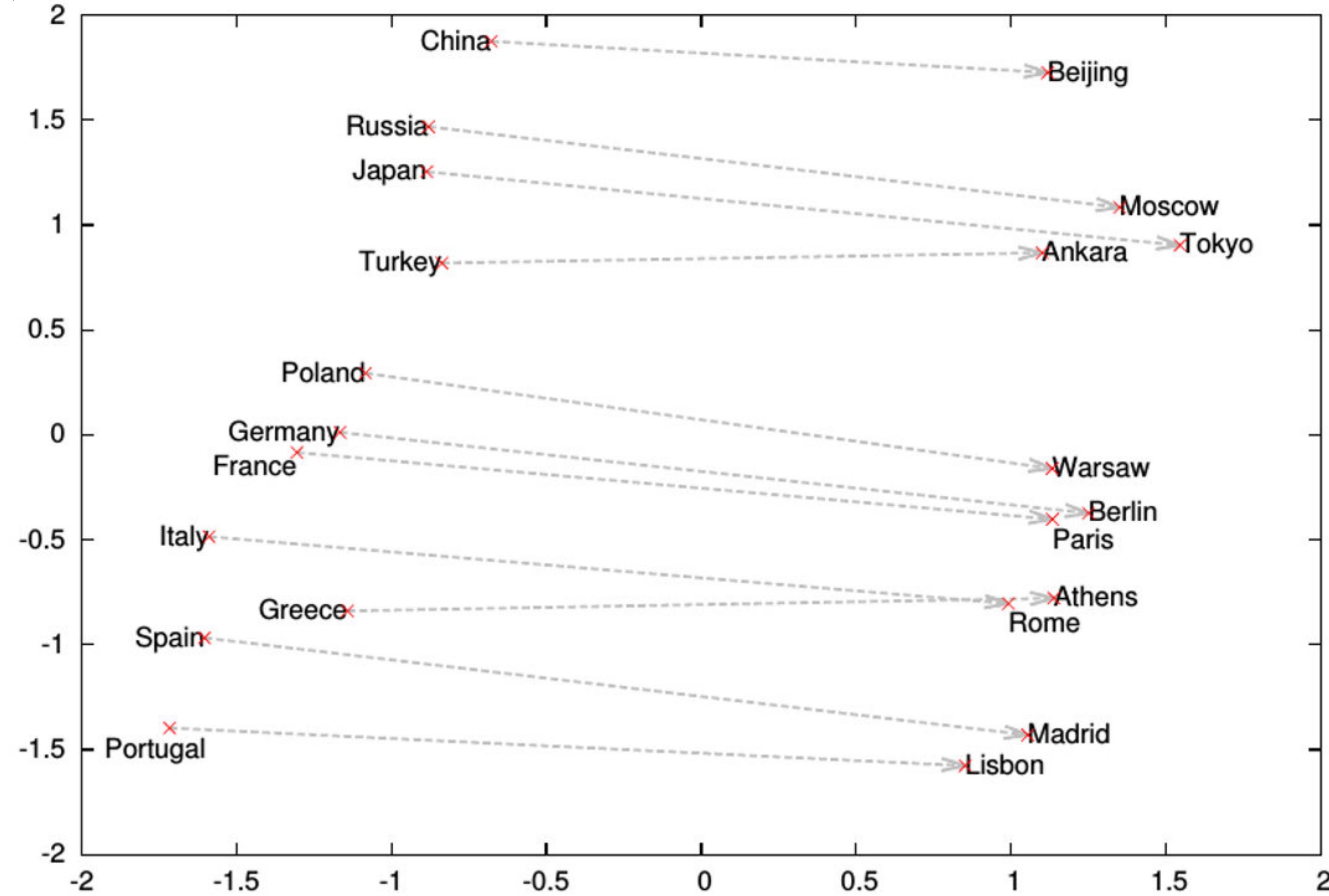
- Known vectors (positions) for:
  - Man
  - Woman
  - King



# RECAP LECTURE 2

## EXAMPLE: WORD2VEC

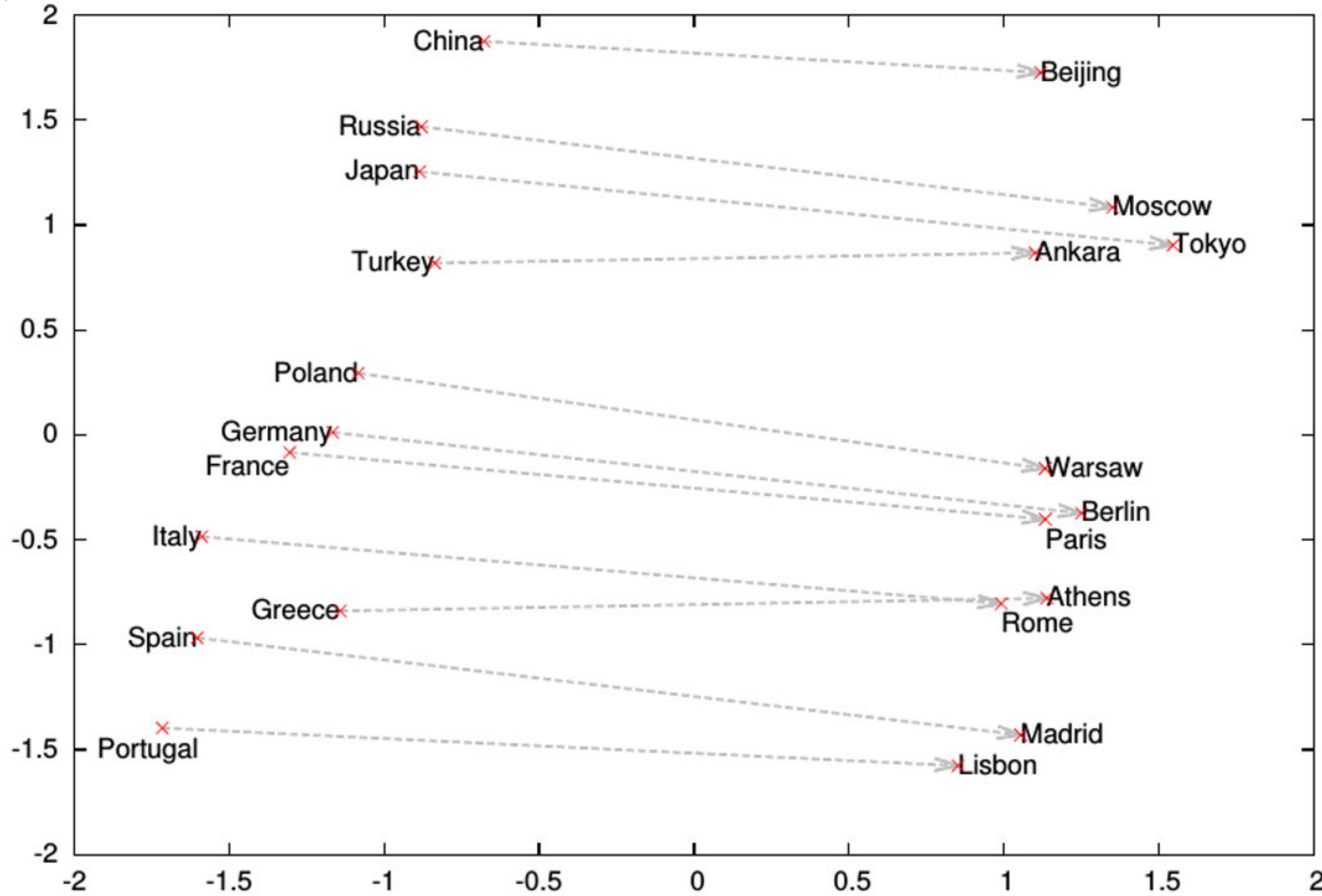
- Known vectors (positions) for:
  - Man
  - Woman
  - King
- Known vectors (differences) for:
  - Man → Woman



# RECAP LECTURE 2

## EXAMPLE: WORD2VEC

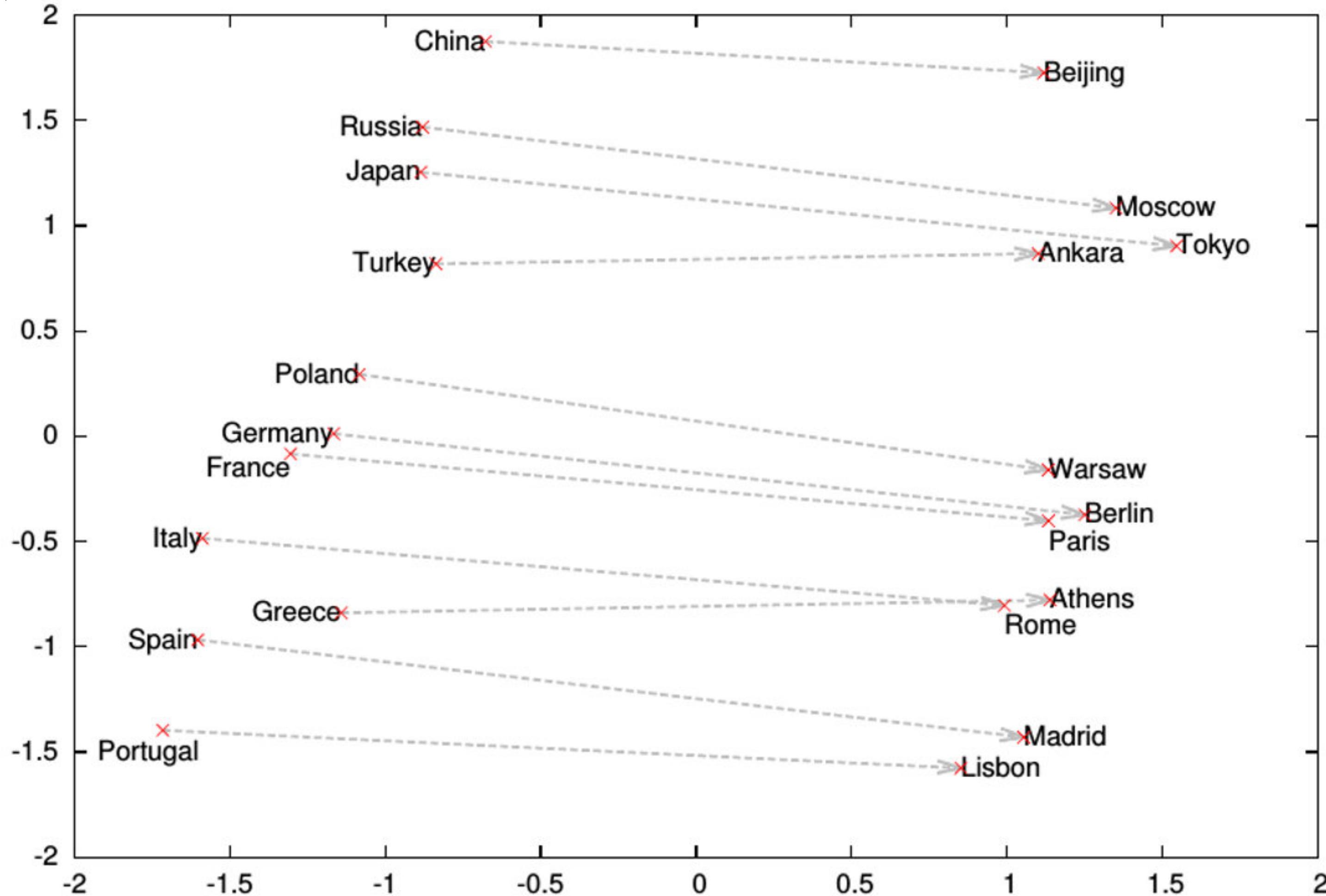
- Known vectors (positions) for:
  - Man
  - Woman
  - King
- Known vectors (differences) for:
  - Man → Woman
- What is the corresponding word for „King“ in the relation „Man → Woman“



# RECAP LECTURE 2

## EXAMPLE: WORD2VEC

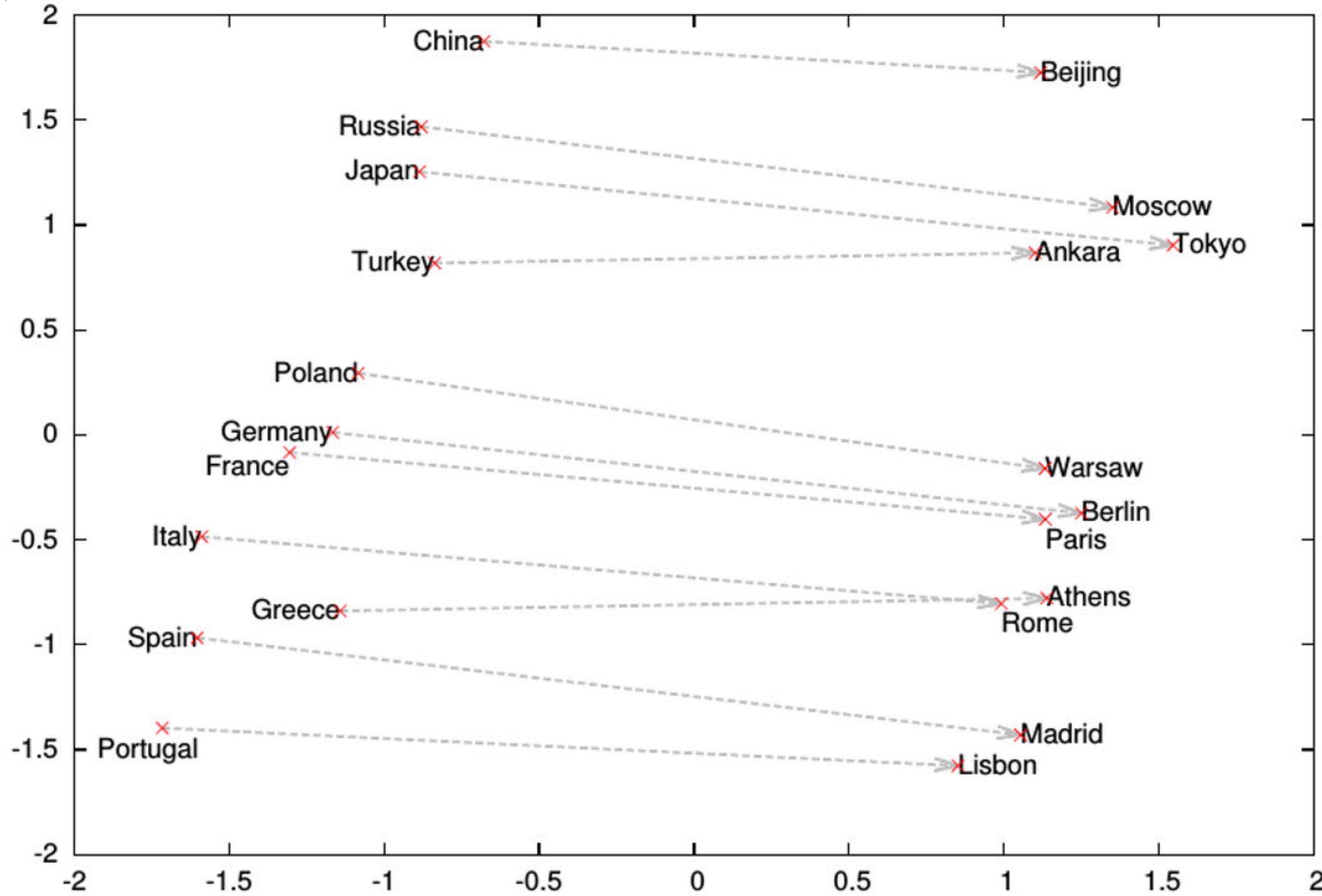
- Known vectors (positions) for:
  - Man
  - Woman
  - King
- Known vectors (differences) for:
  - Man → Woman
- What is the corresponding word for „King“ in the relation „Man → Woman“
  - King → ?



# RECAP LECTURE 2

## EXAMPLE: WORD2VEC

- Known vectors (positions) for:
  - Man
  - Woman
  - King
- Known vectors (differences) for:
  - Man → Woman
- What is the corresponding word for „King“ in the relation „Man → Woman“
  - King → ?
  - King - Man + Woman = Queen

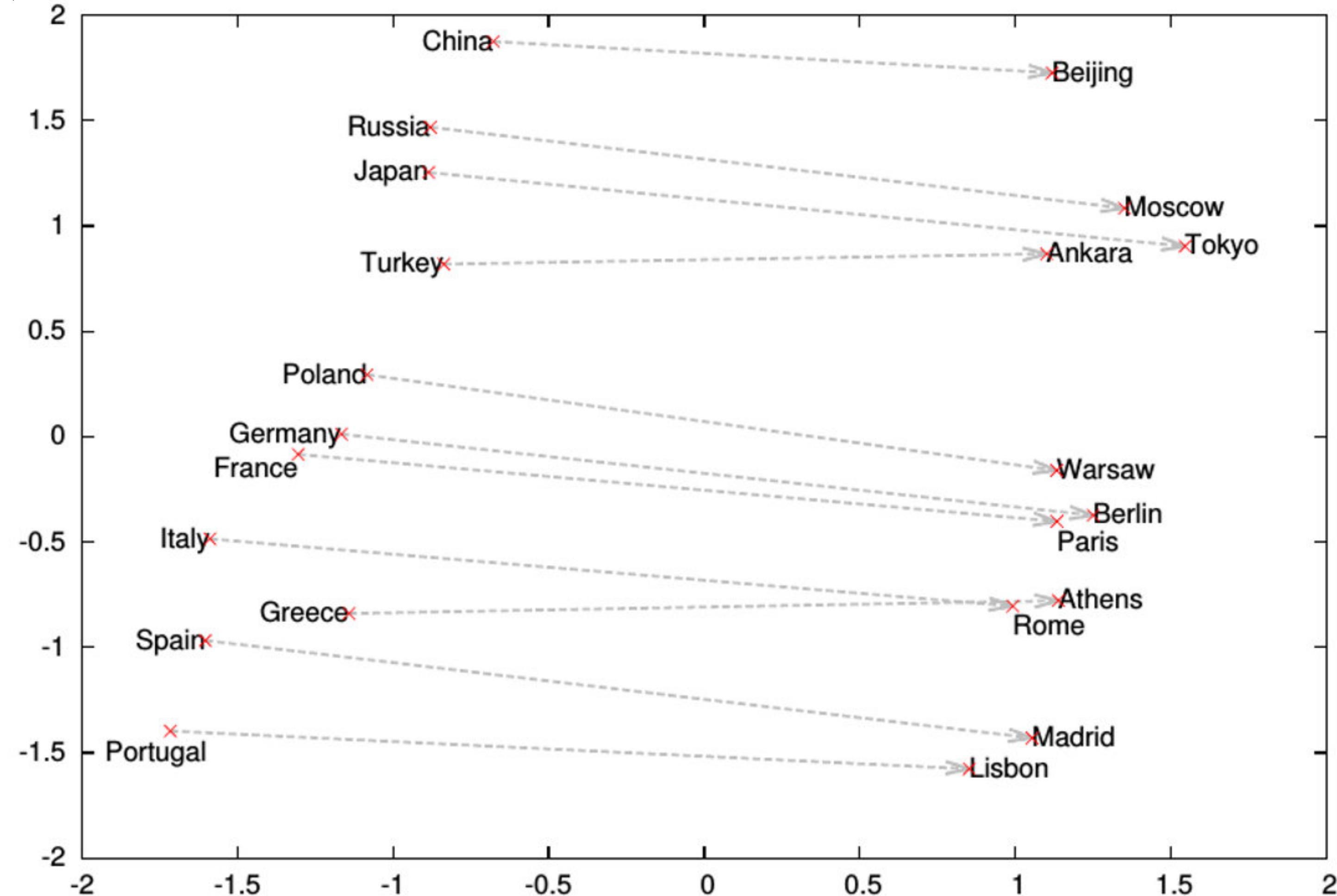


# RECAP LECTURE 2

## EXAMPLE: WORD2VEC

- Known vectors (positions) for:
  - Man
  - Woman
  - King
- Known vectors (differences) for:
  - Man → Woman
- What is the corresponding word for „King“ in the relation „Man → Woman“
  - King → ?
  - King - Man + Woman = Queen

Representing words by vectors in a multi-dimensional vector space



# GET AN EMBEDDING

---



DEPARTMENT OF MANAGEMENT  
AARHUS UNIVERSITY

29. October 2025

Magnus Bender  
Assistant Professor

24



# GET AN EMBEDDING

---

- The OpenAI API provides an embedding endpoint

# GET AN EMBEDDING

---

- The OpenAI API provides an embedding endpoint

```
from openai import OpenAI
OPENAI_API_KEY = "..."
client = OpenAI(api_key=OPENAI_API_KEY)

result = client.embeddings.create(
    model='text-embedding-3-small',
    input="Some text to embed!")
print(result.data[0].embedding)
```

# GET AN EMBEDDING

---

- The OpenAI API provides an embedding endpoint
  - Create a bunch of decimal numbers (vector)
  - This vector represents a string in the embedding space

```
from openai import OpenAI  
  
OPENAI_API_KEY = "..."  
client = OpenAI(api_key=OPENAI_API_KEY)  
  
result = client.embeddings.create(  
    model='text-embedding-3-small',  
    input="Some text to embed!"  
)  
print(result.data[0].embedding)
```

```
[ -0.003715922124683857, 0.007958686910569668,  
-0.0231960229575634, ... , -0.04570453241467476,  
0.02046092413365841, -0.03326955437660217,  
0.0831589326262474 ]
```

# GET AN EMBEDDING

---

- The OpenAI API provides an embedding endpoint
  - Create a bunch of decimal numbers (vector)
  - This vector represents a string in the embedding space
- Do it for all chunks and each question
  - Chunk size matters:

```
from openai import OpenAI
OPENAI_API_KEY = "..."
client = OpenAI(api_key=OPENAI_API_KEY)

result = client.embeddings.create(
    model='text-embedding-3-small',
    input="Some text to embed!")
print(result.data[0].embedding)
```

```
[ -0.003715922124683857, 0.007958686910569668,
-0.0231960229575634, ... , -0.04570453241467476,
0.02046092413365841, -0.03326955437660217,
0.0831589326262474 ]
```

# GET AN EMBEDDING

---

- The OpenAI API provides an embedding endpoint
  - Create a bunch of decimal numbers (vector)
  - This vector represents a string in the embedding space
- Do it for all chunks and each question
  - Chunk size matters:
    - Similar topics per chunk → Meaningful position in embedding space

```
from openai import OpenAI
OPENAI_API_KEY = "..."
client = OpenAI(api_key=OPENAI_API_KEY)

result = client.embeddings.create(
    model='text-embedding-3-small',
    input="Some text to embed!")
print(result.data[0].embedding)
```

```
[ -0.003715922124683857, 0.007958686910569668,
-0.0231960229575634, ... , -0.04570453241467476,
0.02046092413365841, -0.03326955437660217,
0.0831589326262474 ]
```

# GET AN EMBEDDING

---

- The OpenAI API provides an embedding endpoint
  - Create a bunch of decimal numbers (vector)
  - This vector represents a string in the embedding space
- Do it for all chunks and each question
  - Chunk size matters:
    - Similar topics per chunk → Meaningful position in embedding space
    - Mixed topics → No clear position

```
from openai import OpenAI  
  
OPENAI_API_KEY = "..."  
client = OpenAI(api_key=OPENAI_API_KEY)  
  
result = client.embeddings.create(  
    model='text-embedding-3-small',  
    input="Some text to embed!")  
  
print(result.data[0].embedding)
```

```
[ -0.003715922124683857, 0.007958686910569668,  
-0.0231960229575634, ... , -0.04570453241467476,  
0.02046092413365841, -0.03326955437660217,  
0.0831589326262474 ]
```

# RETRIEVE FROM STORED EMBEDDINGS

---

# RETRIEVE FROM STORED EMBEDDINGS

Question Text	Question Embedding
What is Aarhus BSS?	0.0024567306973040104, 0.026274874806404114, 0.04385124146938324, ...

- Assume we have the question and the embedding of the question ↑

# RETRIEVE FROM STORED EMBEDDINGS

Question Text	Question Embedding
What is Aarhus BSS?	0.0024567306973040104, 0.026274874806404114, 0.04385124146938324, ...

- Assume we have the question and the embedding of the question ↑
- In addition, we have all the text chunks and their embeddings in a Excel sheet →

Chunk ID	Chunk Text	Chunk Embedding
1	Aarhus BSS: The School of Business and Social Sciences, Aarhus University. One of the five faculties at AU, which is often just referred to as BSS. Here you will find the Department of Business	-0.003715922124683857, 0.007958686910569668, -0.0231960229575634, ...
2	Development and Technology, the Department of Law, the Department of Psychology and Behavioural	0.026827873662114143, 0.002169022336602211, -0.0106863118708137, ...
3	Sciences, the Department of Political Science, the Department of Management, and the Department of	0.03440544009208679, 0.016993477940559387, -0.0282776243984925, ...
4	Magnus Bender Assistant Professor	0.06151728704571724, 0.03111734427511692, 0.00567943835631129, ...

# RETRIEVE FROM STORED EMBEDDINGS

Question Text	Question Embedding
What is Aarhus BSS?	0.0024567306973040104, 0.026274874806404114, 0.04385124146938324, ...

- Assume we have the question and the embedding of the question ↑
- In addition, we have all the text chunks and their embeddings in a Excel sheet →
- **Task:** Retrieve the most similar chunks to the question

Chunk ID	Chunk Text	Chunk Embedding
1	Aarhus BSS: The School of Business and Social Sciences, Aarhus University. One of the five faculties at AU, which is often just referred to as BSS. Here you will find the Department of Business	-0.003715922124683857, 0.007958686910569668, -0.0231960229575634, ...
2	Development and Technology, the Department of Law, the Department of Psychology and Behavioural Sciences, the Department of Political Science, the Department of Management, and the Department of	0.026827873662114143, 0.002169022336602211, -0.0106863118708137, ...
3	Magnus Bender	0.03440544009208679, 0.016993477940559387, -0.0282776243984925, ...
4	Assistant Professor	0.06151728704571724, 0.03111734427511692, 0.00567943835631129, ...

# RETRIEVE FROM STORED EMBEDDINGS

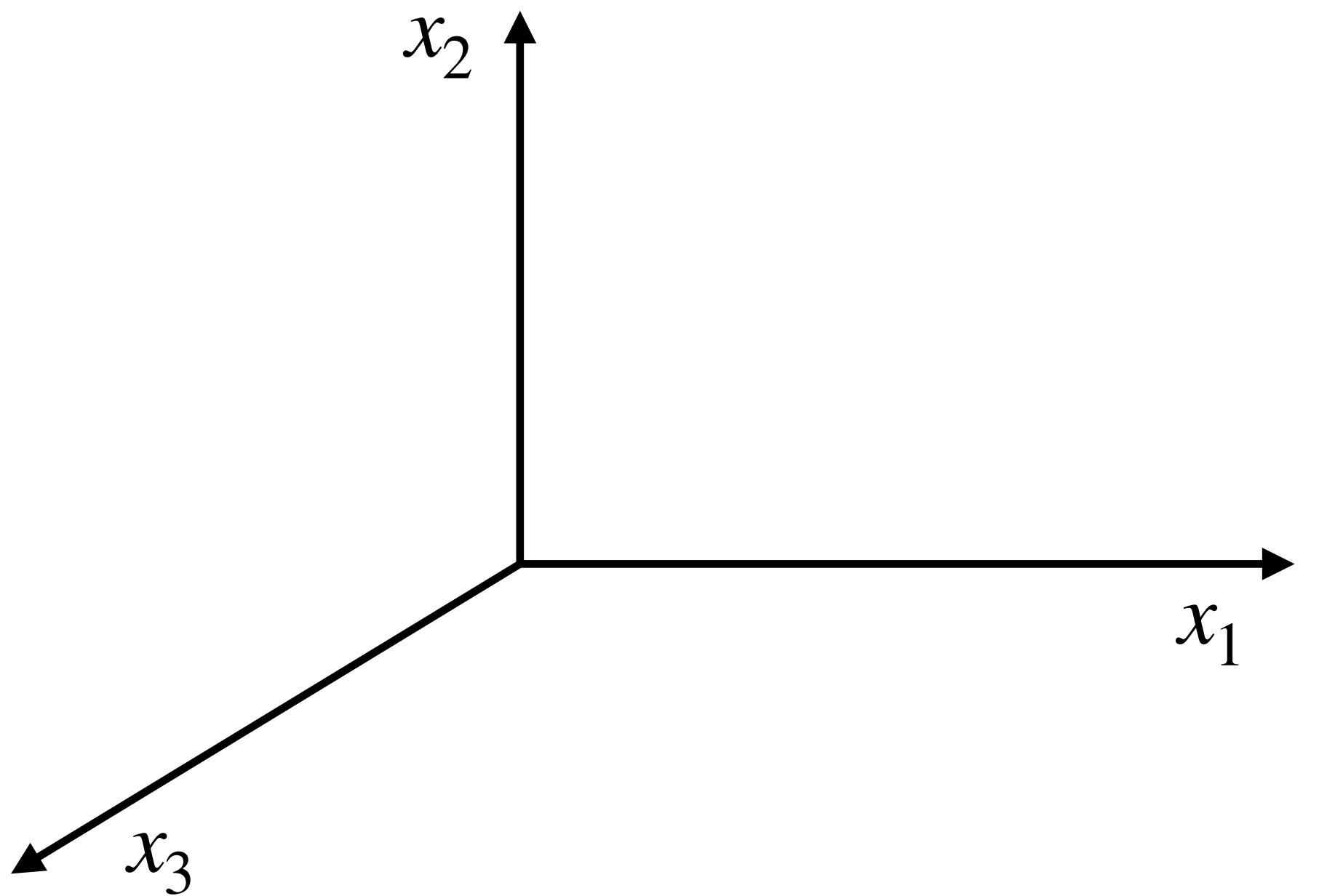
Question Text	Question Embedding
What is Aarhus BSS?	0.0024567306973040104, 0.026274874806404114, 0.04385124146938324, ...

- Assume we have the question and the embedding of the question ↑
- In addition, we have all the text chunks and their embeddings in a Excel sheet →
- **Task:** Retrieve the most similar chunks to the question
- **Method:** Compare question's embedding with each chunks' embedding, take best

Chunk ID	Chunk Text	Chunk Embedding
1	Aarhus BSS: The School of Business and Social Sciences, Aarhus University. One of the five faculties at AU, which is often just referred to as BSS. Here you will find the Department of Business	-0.003715922124683857, 0.007958686910569668, -0.0231960229575634, ...
2	Development and Technology, the Department of Law, the Department of Psychology and Behavioural	0.026827873662114143, 0.002169022336602211, -0.0106863118708137, ...
3	Sciences, the Department of Political Science, the Department of Management, and the Department of	0.03440544009208679, 0.016993477940559387, -0.0282776243984925, ...
4	Magnus Bender Assistant Professor	0.06151728704571724, 0.03111734427511692, 0.00567943835631129, ...

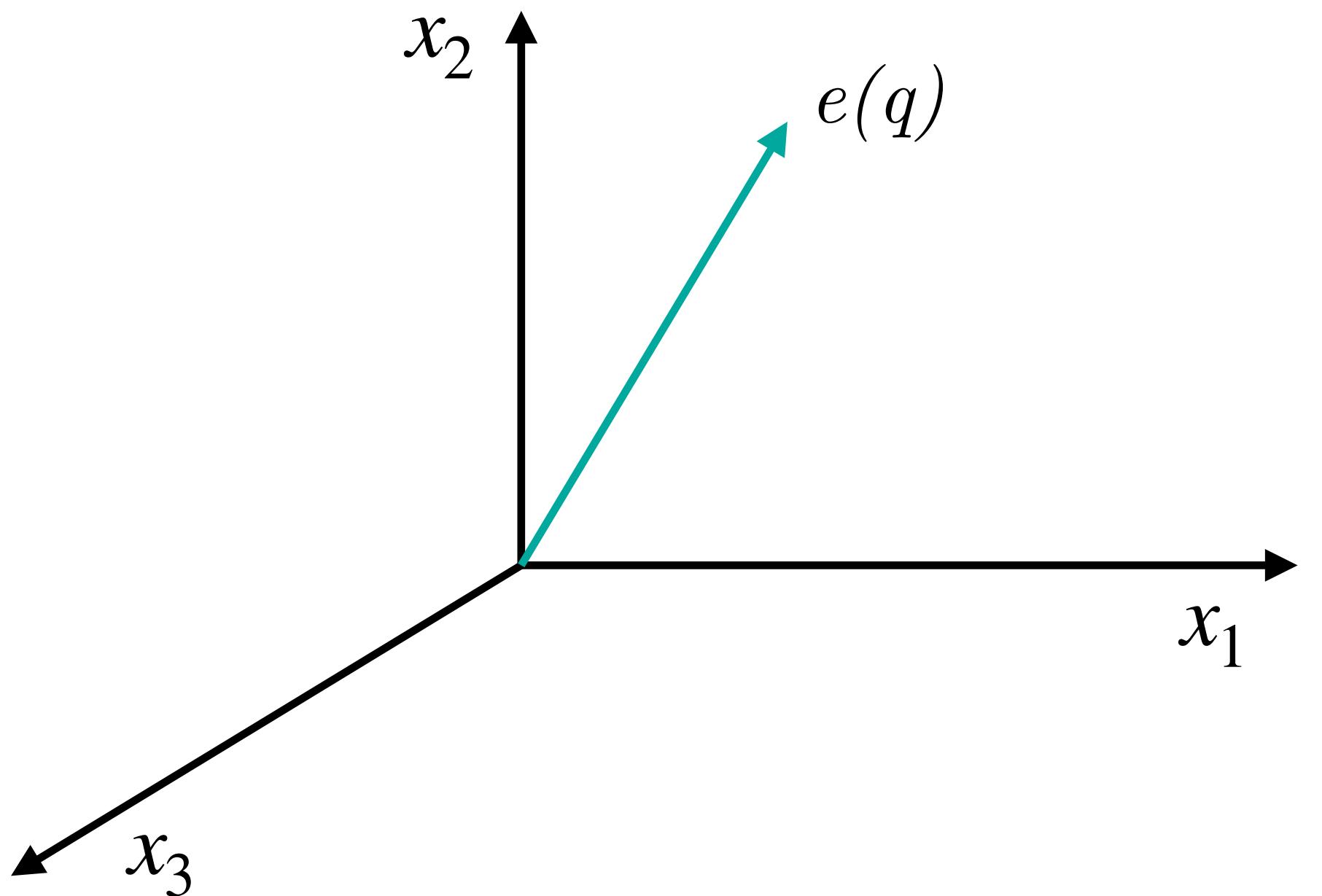
# SIMILARITY OF (EMBEDDING) VECTORS

---



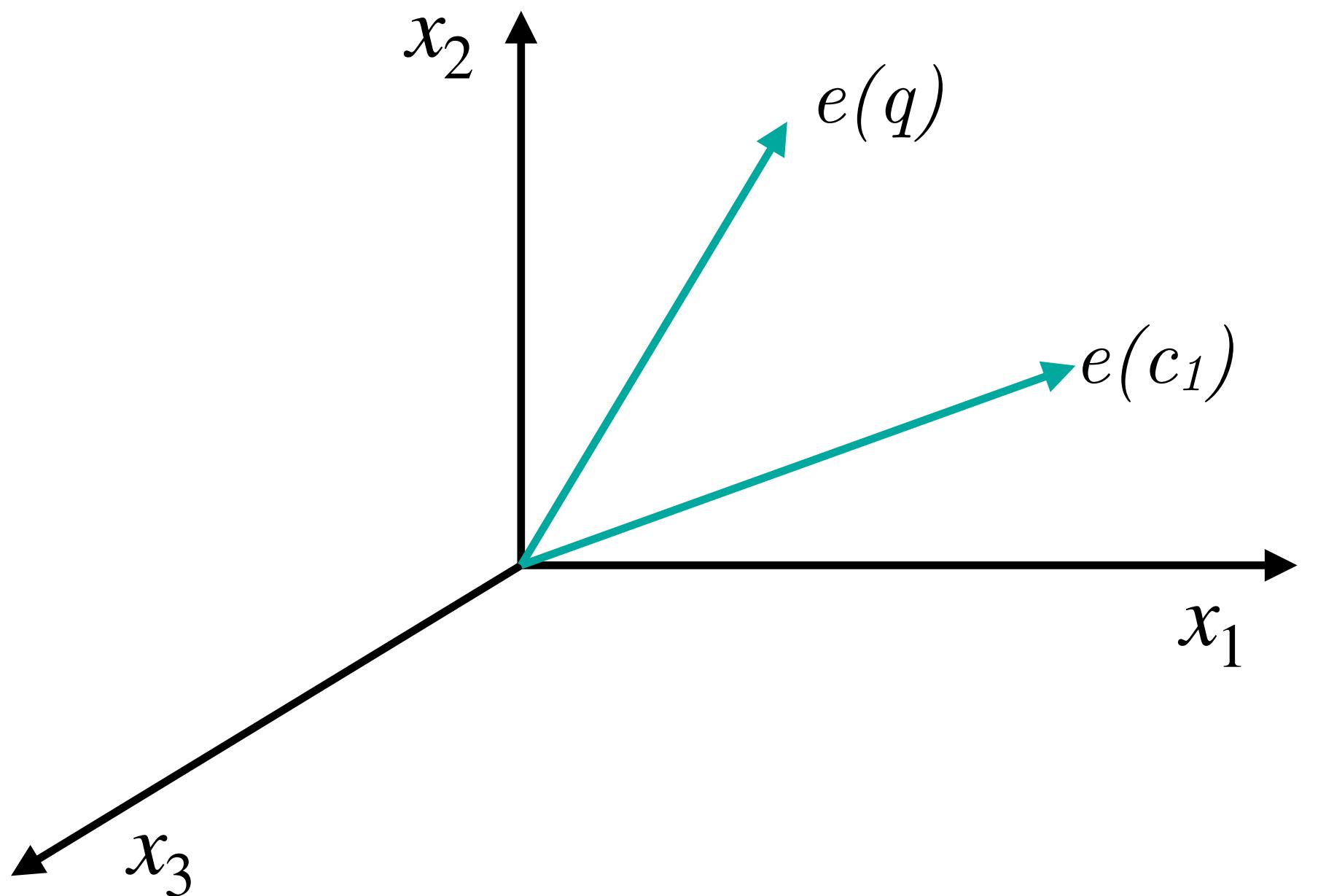
# SIMILARITY OF (EMBEDDING) VECTORS

---



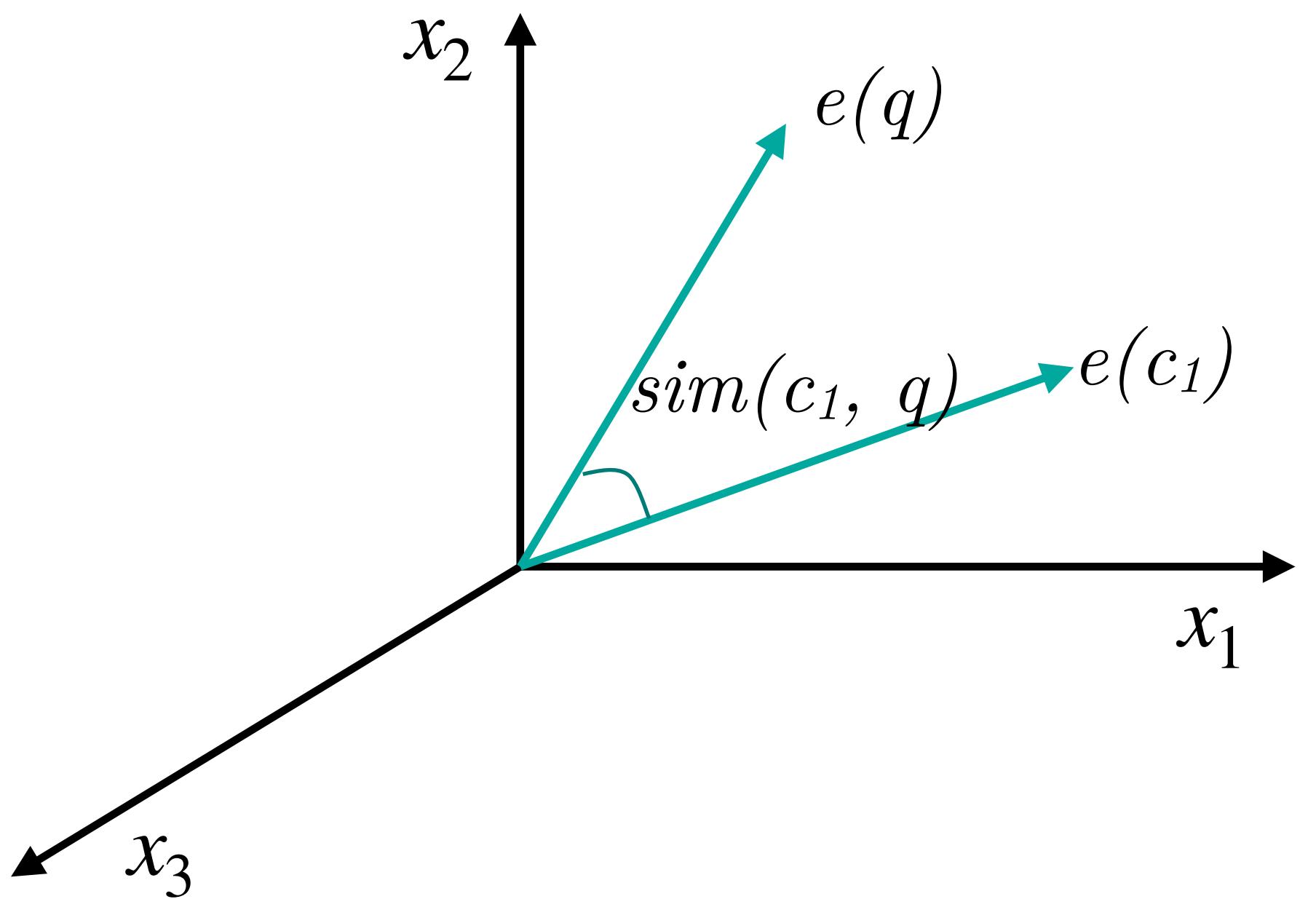
# SIMILARITY OF (EMBEDDING) VECTORS

---



# SIMILARITY OF (EMBEDDING) VECTORS

---

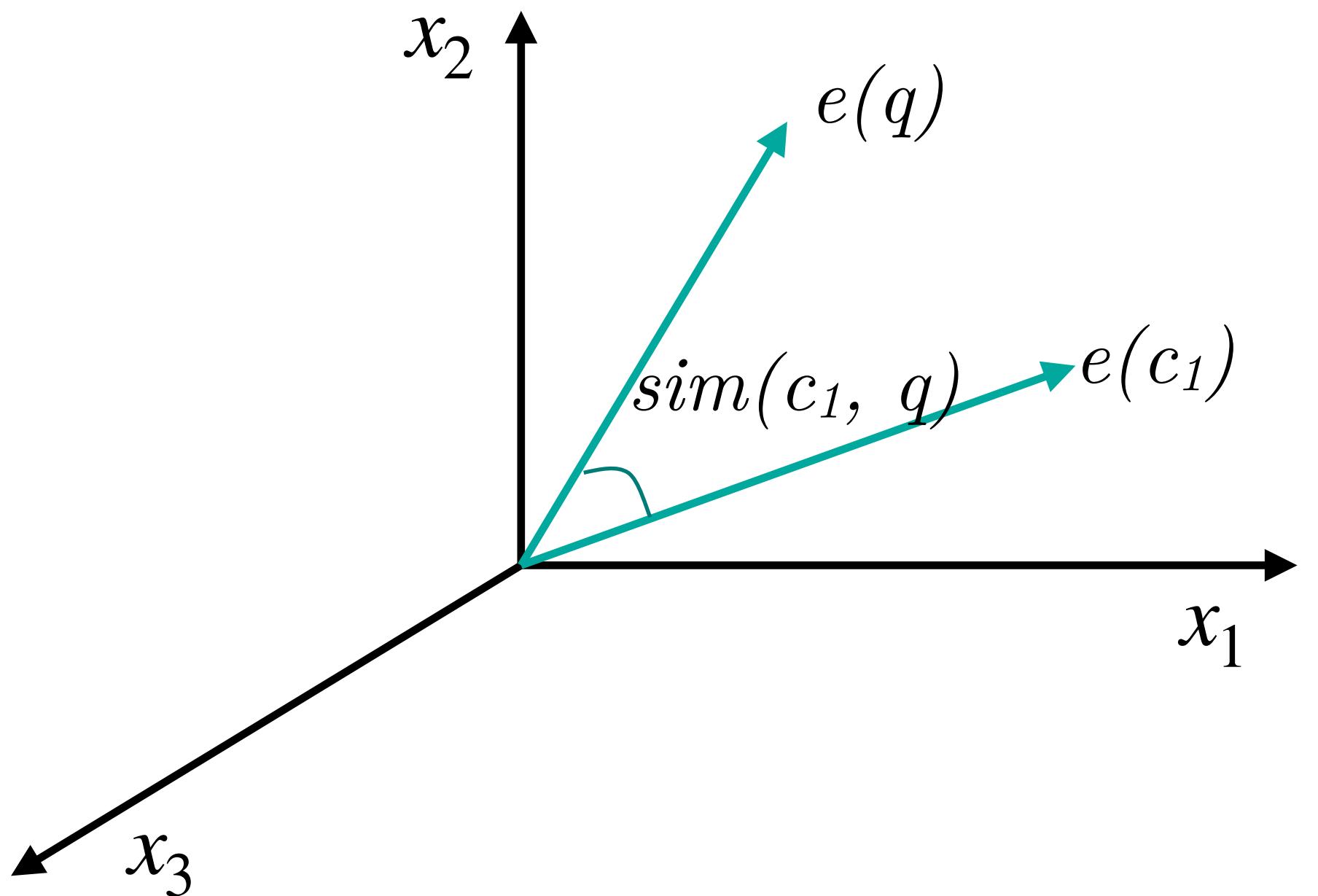


# SIMILARITY OF (EMBEDDING) VECTORS

---

$$\text{sim}(c_1, q) = \cos(e(c_1), e(q))$$

- Use the cosine-similarity, i.e., the angle between two vectors



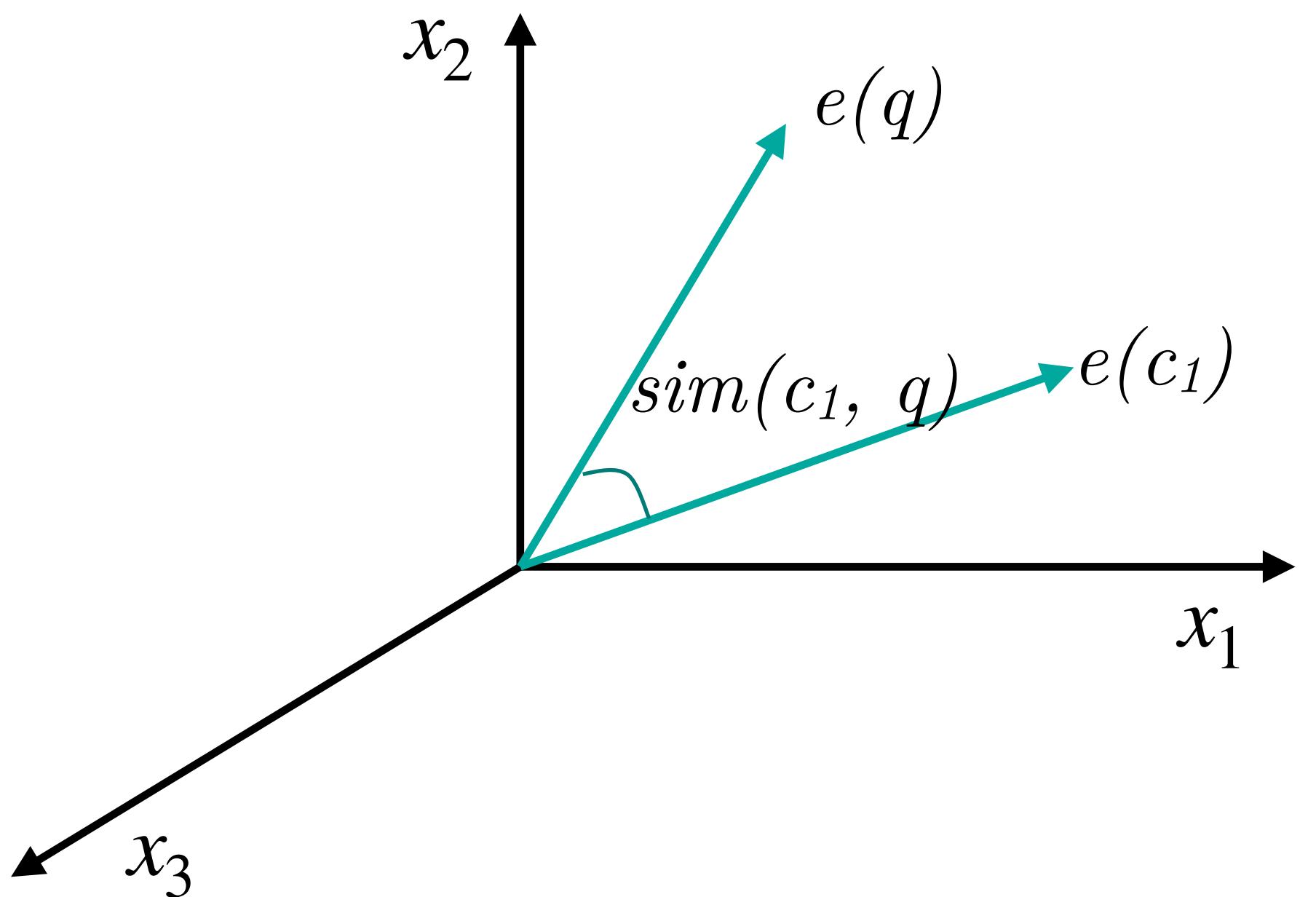
# SIMILARITY OF (EMBEDDING) VECTORS

---

$$\text{sim}(c_1, q) = \cos(e(c_1), e(q))$$

$$\cos(\vec{v}, \vec{w}) = \frac{\vec{v} \cdot \vec{w}}{\|\vec{v}\| \|\vec{w}\|} = \frac{\sum_{i=1}^K v_i w_i}{\sqrt{\sum_{i=1}^K v_i^2} \sqrt{\sum_{i=1}^K w_i^2}}$$

- Use the cosine-similarity, i.e., the angle between two vectors



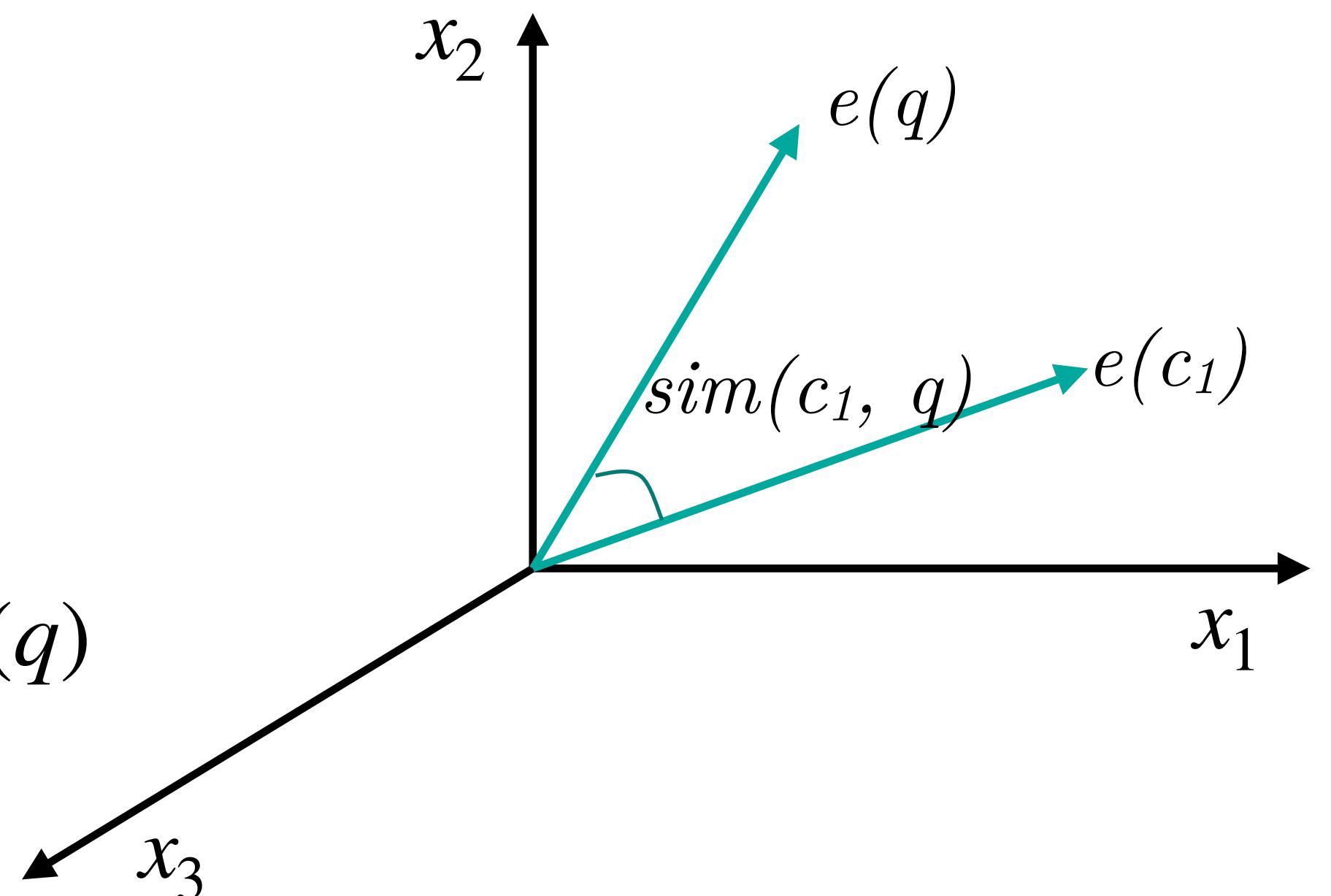
# SIMILARITY OF (EMBEDDING) VECTORS

---

$$\text{sim}(c_1, q) = \cos(e(c_1), e(q))$$

$$\cos(\vec{v}, \vec{w}) = \frac{\vec{v} \cdot \vec{w}}{\|\vec{v}\| \|\vec{w}\|} = \frac{\sum_{i=1}^K v_i w_i}{\sqrt{\sum_{i=1}^K v_i^2} \sqrt{\sum_{i=1}^K w_i^2}}$$

- Use the cosine-similarity, i.e., the angle between two vectors
- Calculate it between the embeddings of each chunk  $e(c_1), e(c_2), \dots, e(c_n)$  and the embedding of the question  $e(q)$



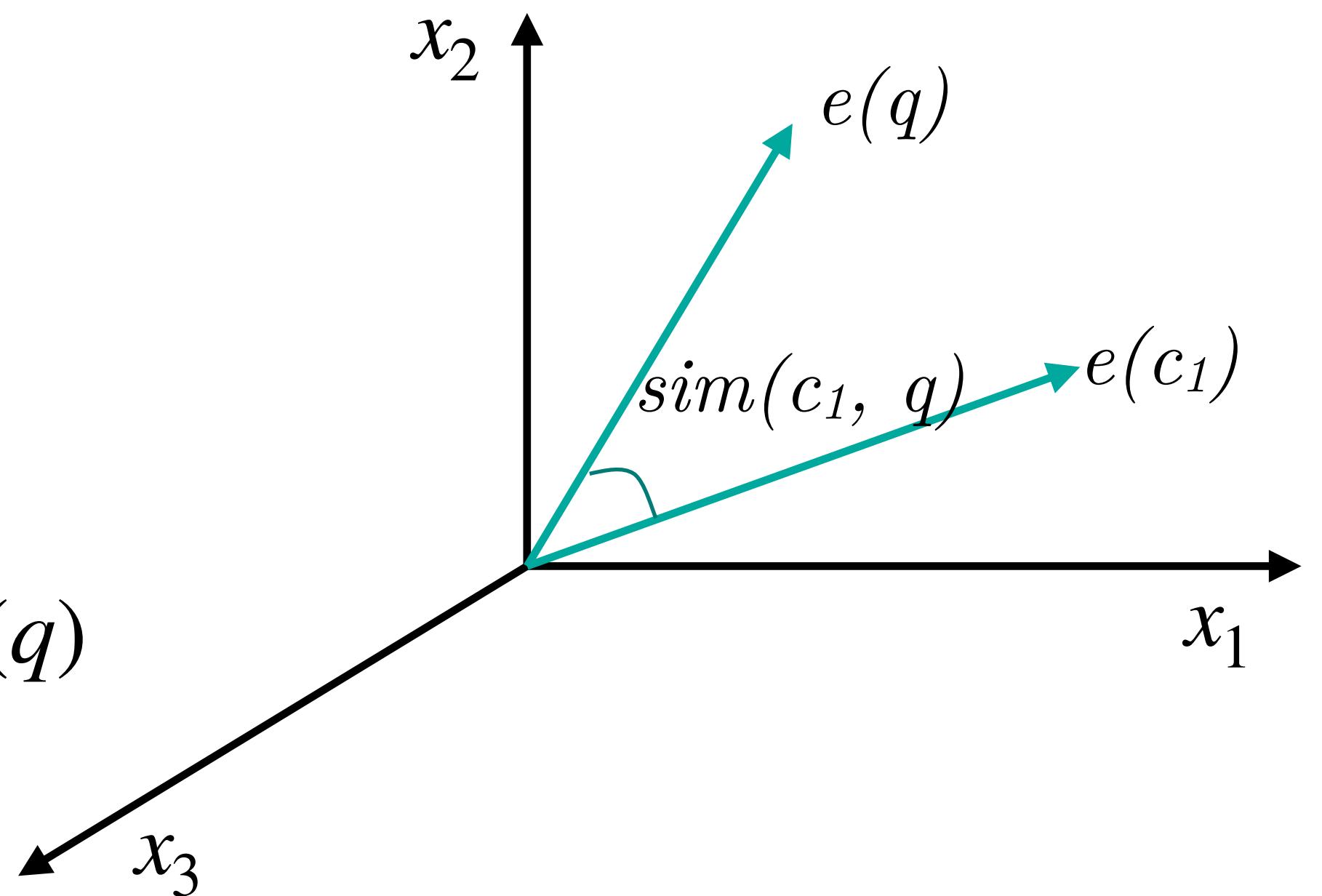
# SIMILARITY OF (EMBEDDING) VECTORS

---

$$\text{sim}(c_1, q) = \cos(e(c_1), e(q))$$

$$\cos(\vec{v}, \vec{w}) = \frac{\vec{v} \cdot \vec{w}}{\|\vec{v}\| \|\vec{w}\|} = \frac{\sum_{i=1}^K v_i w_i}{\sqrt{\sum_{i=1}^K v_i^2} \sqrt{\sum_{i=1}^K w_i^2}}$$

- Use the cosine-similarity, i.e., the angle between two vectors
- Calculate it between the embeddings of each chunk  $e(c_1), e(c_2), \dots, e(c_n)$  and the embedding of the question  $e(q)$
- Take the best  $k$  chunks of  $\text{sim}(c_1, q), \text{sim}(c_2, q), \dots, \text{sim}(c_n, q)$  for augmentation



# RAG: SUMMARIZED

---



# RAG: SUMMARIZED

## — A. Initialize with a dataset

# RAG: SUMMARIZED

---

- A. Initialize with a dataset
  - 1. Split dataset into chunks

# RAG: SUMMARIZED

---

- A. Initialize with a dataset
  - 1. Split dataset into chunks
  - 2. Create embedding for each chunk

# RAG: SUMMARIZED

---

- A. Initialize with a dataset
  - 1. Split dataset into chunks
  - 2. Create embedding for each chunk
  - 3. Store embeddings and chunks

# RAG: SUMMARIZED

- - A. Initialize with a dataset
    - 1. Split dataset into chunks
    - 2. Create embedding for each chunk
    - 3. Store embeddings and chunks
  - B. Augment a question

# RAG: SUMMARIZED

- - A. Initialize with a dataset
    - 1. Split dataset into chunks
    - 2. Create embedding for each chunk
    - 3. Store embeddings and chunks
  - B. Augment a question
    - 1. Create embedding of questions

# RAG: SUMMARIZED

---

- A. Initialize with a dataset
  - 1. Split dataset into chunks
  - 2. Create embedding for each chunk
  - 3. Store embeddings and chunks
- B. Augment a question
  - 1. Create embedding of questions
  - 2. Retrieve most similar chunks to the question, use the embeddings, e.g.:

# RAG: SUMMARIZED

- - A. Initialize with a dataset
    - 1. Split dataset into chunks
    - 2. Create embedding for each chunk
    - 3. Store embeddings and chunks
  - B. Augment a question
    - 1. Create embedding of questions
    - 2. Retrieve most similar chunks to the question, use the embeddings, e.g.:
      - ▶ Calculate similarity of embedding of question with each chunk's embedding
      - ▶ Take the top  $k$  or all above a similarity threshold

# RAG: SUMMARIZED

---

- A. Initialize with a dataset
  - 1. Split dataset into chunks
  - 2. Create embedding for each chunk
  - 3. Store embeddings and chunks
- B. Augment a question
  - 1. Create embedding of questions
  - 2. Retrieve most similar chunks to the question, use the embeddings, e.g.:
    - ▶ Calculate similarity of embedding of question with each chunk's embedding
    - ▶ Take the top  $k$  or all above a similarity threshold
  - 3. Augment the question with the most similar (relevant) chunks

# RAG: SUMMARIZED

- - A. Initialize with a dataset
    - 1. Split dataset into chunks
    - 2. Create embedding for each chunk
    - 3. Store embeddings and chunks
  - B. Augment a question
    - 1. Create embedding of questions
    - 2. Retrieve most similar chunks to the question, use the embeddings, e.g.:
      - ▶ Calculate similarity of embedding of question with each chunk's embedding
      - ▶ Take the top  $k$  or all above a similarity threshold
    - 3. Augment the question with the most similar (relevant) chunks
    - 4. Let *general LLM* generate an answer to question

# RAG: SUMMARIZED

- A. Initialize with a dataset
  - 1. Split dataset into chunks
  - 2. Create embedding for each chunk
  - 3. Store embeddings and chunks
- B. Augment a question
  - 1. Create embedding of questions
  - 2. Retrieve most similar chunks to the question, use the embeddings, e.g.:
    - ▶ Calculate similarity of embedding of question with each chunk's embedding
    - ▶ Take the top  $k$  or all above a similarity threshold
  - 3. Augment the question with the most similar (relevant) chunks
  - 4. Let *general LLM* generate an answer to question

→ We will implement it in the next tutorial

# FUNCTION CALLING

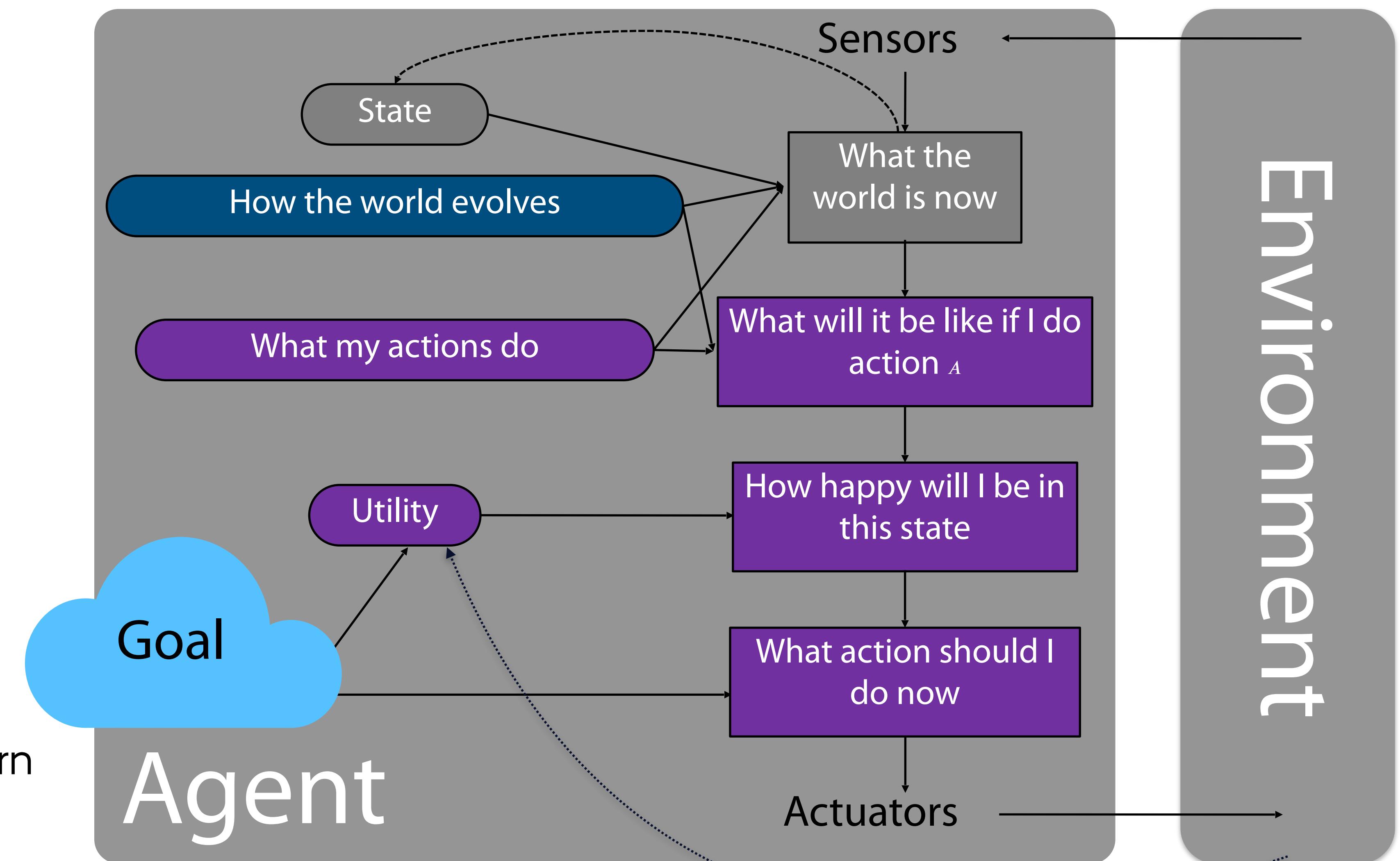
---

- Wiring a toolbox to GPT

# RECAP LECTURE 2

## AI AGENT

- Intelligent systems, but not necessarily *intelligent* in a human sense
- Agents
  - ... have goals
  - ... have a perception of their environment (sensors)
  - ... can change their environment (actuators)
  - ... plan their actions
  - ... update theirs goals → learn during runtime



# READINGS: EQUIPPING LANGUAGE MODELS WITH TOOLS

How much was the included change in fair value of the company's servicing asset included in its servicing fees?  
**GreenSky, Inc. NOTES TO CONSOLIDATED FINANCIAL STATEMENTS (Continued) (United States Dollars in thousands, except per share...**  
`{"header": ["", "", "Year Ended December 31", ""], "rows": [{"", "2019", "2018", "2017"}, {"Merchant fees", "$361,755", "$297,776", "$234,548"}], ...}`

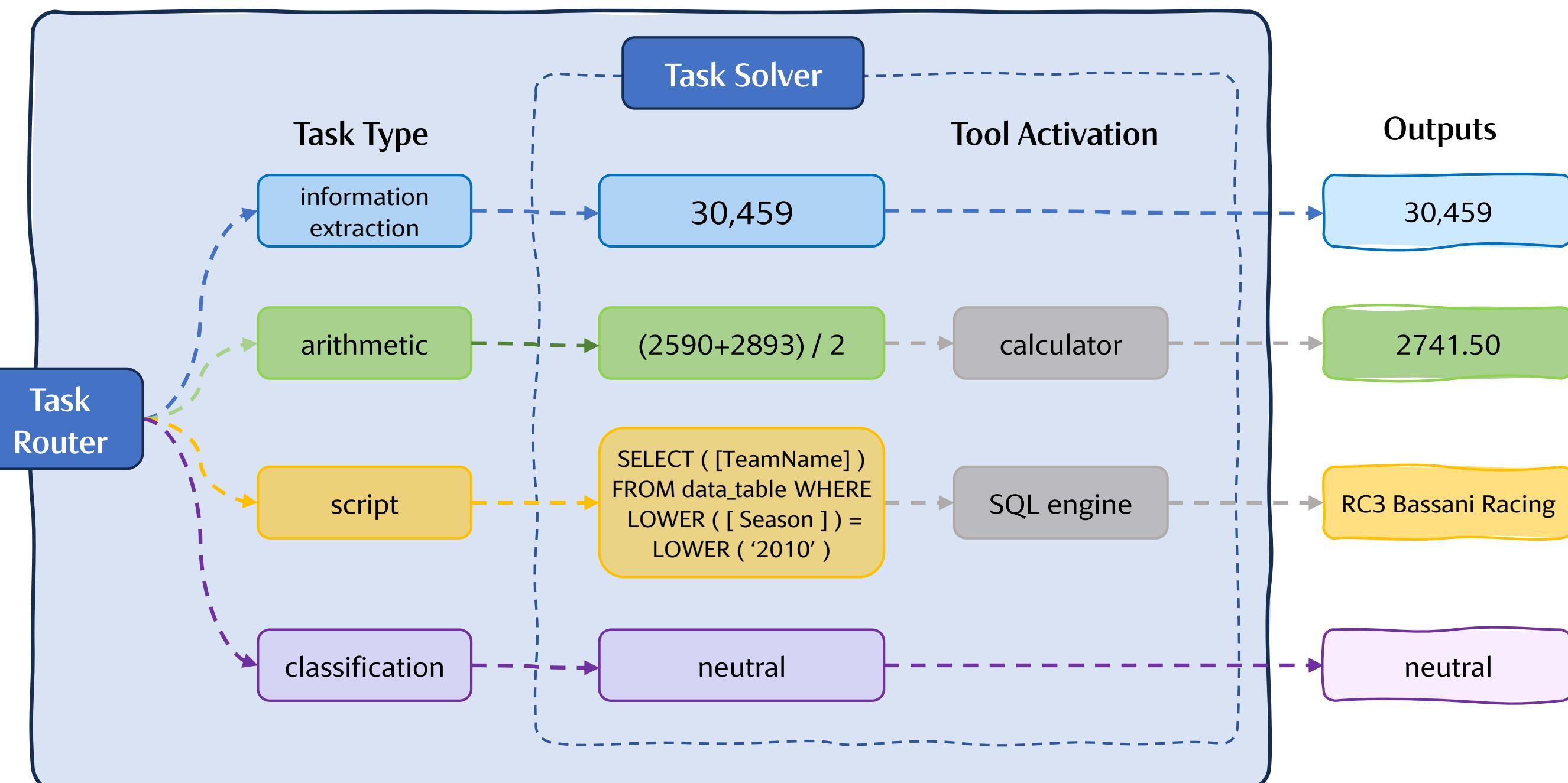
What is the average of External Systems Hardware gross profit?  
The Systems gross profit margin decrease year to year was driven by the mix away from IBM Z and margin declines in Power Systems and Storage Systems...  
`{"header": ["($ in millions)", "", "", ""], "rows": [{"For the year ended December 31", "2018", "2017", "Yr.-to-Yr. Percent/ Margin Change"}, ...]}`

Which team started 2010?  
`{"header": ["Season", "Series", "Team Name", "Races", "Poles", "Wins", "Points", "FinalPlacing"], "rows": [{"2002", "FormulaRenault 2.0 Brazil", ...}]} ...`

Determine the sentiment of the following.  
Metsa-Botnia will finance the payment of dividends , the repayment of capital and the repurchase of its own shares with the funds deriving from its ...



RAVEN



# FUNCTION CALLING AND THE OPEN AI API: CREATING A FUNCTION

---

- Example use-case:

# FUNCTION CALLING AND THE OPEN AI API: CREATING A FUNCTION

---

- Example use-case:
  - Check which employee is available for a meeting

# FUNCTION CALLING AND THE OPEN AI API: CREATING A FUNCTION

---

- Example use-case:
  - Check which employee is available for a meeting
  - Provide a simple calendar lookup function to ChatGPT

# FUNCTION CALLING AND THE OPEN AI API: CREATING A FUNCTION

---

- Example use-case:
  - Check which employee is available for a meeting
  - Provide a simple calendar lookup function to ChatGPT

```
employee_calendar = {
    "James" : [("Monday", 10), ("Monday", 11), ("Tuesday", 11)],
    "William" : [("Monday", 9), ("Monday", 10), ("Tuesday", 11)],
    "Mary" : [("Monday", 9), ("Tuesday", 10), ("Tuesday", 12)],
    "Elizabeth" : [("Monday", 9), ("Tuesday", 10), ("Tuesday", 12)],
}
```

# FUNCTION CALLING AND THE OPEN AI API: CREATING A FUNCTION

---

- Example use-case:
  - Check which employee is available for a meeting
  - Provide a simple calendar lookup function to ChatGPT

```
employee_calendar = {
    "James" : [("Monday", 10), ("Monday", 11), ("Tuesday", 11)],
    "William" : [("Monday", 9), ("Monday", 10), ("Tuesday", 11)],
    "Mary" : [("Monday", 9), ("Tuesday", 10), ("Tuesday", 12)],
    "Elizabeth" : [("Monday", 9), ("Tuesday", 10), ("Tuesday", 12)],
}

def is_employee_available(name:str, day:str, hour:int) -> bool:
    if name in employee_calendar:
        for d, h in employee_calendar[name]:
            if d == day and h == hour:
                return False
    return True
else:
    return False
```

# FUNCTION CALLING AND THE OPEN AI API: TELLING CHATGPT ABOUT A FUNCTION

—

*def is\_employee\_available(name:str, day:str, hour:int)*

# FUNCTION CALLING AND THE OPEN AI API: TELLING CHATGPT ABOUT A FUNCTION

*def is\_employee\_available(name:str, day:str, hour:int)*

```
tools = [{}  
"type": "function",  
"name": "is_employee_available",  
"description": "Checks if an employee is available at a current day and hour.",  
"parameters": {  
    "type": "object",  
    "properties": {  
        ...  
    },  
    "required": ["name", "day", "hour"],  
    "additionalProperties": False  
},  
"strict": True  
}]
```

# FUNCTION CALLING AND THE OPEN AI API: TELLING CHATGPT ABOUT A FUNCTION

```
def is_employee_available(name:str, day:str, hour:int)
```

```
tools = [{}  
    "type": "function",  
    "name": "is_employee_available",  
    "description": "Checks if an employee is available at a current day and hour.",  
    "parameters": {  
        "type": "object",  
        "properties": {  
            ...  
        },  
        "required": ["name", "day", "hour"],  
        "additionalProperties": False  
    },  
    "strict": True  
}]
```



```
    "name": {  
        "type": "string",  
        "enum": list(employee_calendar.keys()),  
        "description": "The name of the employee."  
    },  
    "day": {  
        "type": "string",  
        "enum": ["Monday", "Tuesday", ...],  
        "description": "The day to check."  
    },  
    "hour": {  
        "type": "integer",  
        "description": "The hour to check, e.g.: 10, 11  
    }]
```

29. October 2025

# DETERMINE FUNCTION CALLS AND EXECUTE THEM

---

```
messages = [{"role": "user", "content":  
"We will have a meeting on Tuesday at 11,  
which employees are free to join  
the meeting?}]
```

# DETERMINE FUNCTION CALLS AND EXECUTE THEM

```
messages = [{"role": "user", "content":  
"We will have a meeting on Tuesday at 11,  
which employees are free to join  
the meeting?}]
```

```
response = client.responses.create(  
    model="gpt-5-nano",  
    tools=tools,  
    input=messages,  
)  
messages.extend(response.output)
```

# DETERMINE FUNCTION CALLS AND EXECUTE THEM

```
messages = [{"role": "user", "content":  
"We will have a meeting on Tuesday at 11,  
which employees are free to join  
the meeting?}]
```

```
response = client.responses.create(  
    model="gpt-5-nano",  
    tools=tools,  
    input=messages,  
)  
messages.extend(response.output)  
  
for item in response.output:
```

# DETERMINE FUNCTION CALLS AND EXECUTE THEM

```
messages = [{"role": "user", "content":  
"We will have a meeting on Tuesday at 11,  
which employees are free to join  
the meeting?}]
```

```
response = client.responses.create(  
    model="gpt-5-nano",  
    tools=tools,  
    input=messages,  
)  
messages.extend(response.output)  
  
for item in response.output:  
    if item.type == "function_call":  
        if item.name == "is_employee_available":
```

# DETERMINE FUNCTION CALLS AND EXECUTE THEM

```
messages = [{"role": "user", "content": "We will have a meeting on Tuesday at 11, which employees are free to join the meeting?"}]
```

```
response = client.responses.create(  
    model="gpt-5-nano",  
    tools=tools,  
    input=messages,  
)  
messages.extend(response.output)  
  
for item in response.output:  
    if item.type == "function_call":  
        if item.name == "is_employee_available":  
  
            args = json.loads(item.arguments)  
            availability = is_employee_available(**args)  
            print(f"is_employee_available({args}): {availability}")
```

# DETERMINE FUNCTION CALLS AND EXECUTE THEM

```
messages = [{"role": "user", "content": "We will have a meeting on Tuesday at 11, which employees are free to join the meeting?"}]
```

```
response = client.responses.create(  
    model="gpt-5-nano",  
    tools=tools,  
    input=messages,  
)  
messages.extend(response.output)  
  
for item in response.output:  
    if item.type == "function_call":  
        if item.name == "is_employee_available":  
  
            args = json.loads(item.arguments)  
            availability = is_employee_available(**args)  
            print(f"is_employee_available({args}): {availability}")
```

# DETERMINE FUNCTION CALLS AND EXECUTE THEM

```
is_employee_available({  
    'name': 'James',  
    'day': 'Tuesday',  
    'hour': 11  
}): False  
is_employee_available({  
    'name': 'William',  
    'day': 'Tuesday',  
    'hour': 11  
}): False  
is_employee_available({  
    'name': 'Mary',  
    'day': 'Tuesday',  
    'hour': 11  
}): True  
is_employee_available({  
    'name': 'Elizabeth',  
    'day': 'Tuesday',  
    'hour': 11  
}): True
```

```
messages = [{"role": "user", "content": "We will have a meeting on Tuesday at 11, which employees are free to join the meeting?"}]
```

# DETERMINE FUNCTION CALLS AND EXECUTE THEM

```
response = client.responses.create(  
    model="gpt-5-nano",  
    tools=tools,  
    input=messages,  
)  
messages.extend(response.output)  
  
for item in response.output:  
    if item.type == "function_call":  
        if item.name == "is_employee_available":  
  
            args = json.loads(item.arguments)  
            availability = is_employee_available(**args)  
            print(f"is_employee_available({args}): {availability}")
```

```
is_employee_available({  
    'name': 'James',  
    'day': 'Tuesday',  
    'hour': 11  
}): False  
is_employee_available({  
    'name': 'William',  
    'day': 'Tuesday',  
    'hour': 11  
}): False  
is_employee_available({  
    'name': 'Mary',  
    'day': 'Tuesday',  
    'hour': 11  
}): True  
is_employee_available({  
    'name': 'Elizabeth',  
    'day': 'Tuesday',  
    'hour': 11  
}): True
```

```
messages = [{"role": "user", "content": "We will have a meeting on Tuesday at 11, which employees are free to join the meeting?"}]
```

# DETERMINE FUNCTION CALLS AND EXECUTE THEM

```
response = client.responses.create(  
    model="gpt-5-nano",  
    tools=tools,  
    input=messages,  
)  
messages.extend(response.output)  
  
for item in response.output:  
    if item.type == "function_call":  
        if item.name == "is_employee_available":  
  
            args = json.loads(item.arguments)  
            availability = is_employee_available(**args)  
            print(f"is_employee_available({args}): {availability}")  
  
            messages.append(  
                {"type": "function_call_output",  
                 "call_id": item.call_id,  
                 "output": "yes" if availability else "no"}  
)
```

```
is_employee_available({  
    'name': 'James',  
    'day': 'Tuesday',  
    'hour': 11  
}): False  
is_employee_available({  
    'name': 'William',  
    'day': 'Tuesday',  
    'hour': 11  
}): False  
is_employee_available({  
    'name': 'Mary',  
    'day': 'Tuesday',  
    'hour': 11  
}): True  
is_employee_available({  
    'name': 'Elizabeth',  
    'day': 'Tuesday',  
    'hour': 11  
}): True
```

# FUNCTION CALLING AND THE OPEN AI API: GETTING THE ANSWER

---

```
response = client.responses.create(  
    model="gpt-5-nano",  
    instructions="Respond only with a list  
    of employees that are available for  
    the meeting in question.",  
    tools=tools,  
    input=messages,  
    # may use structured outputs etc.  
)  
  
print(response.output_text)
```

- Making tools (self defined local Python functions) available to ChatGPT

# FUNCTION CALLING AND THE OPEN AI API: GETTING THE ANSWER

---

```
response = client.responses.create(  
    model="gpt-5-nano",  
    instructions="Respond only with a list  
    of employees that are available for  
    the meeting in question.",  
    tools=tools,  
    input=messages,  
    # may use structured outputs etc.  
)  
  
print(response.output_text)
```

Mary  
Elizabeth



DEPARTMENT OF MANAGEMENT  
AARHUS UNIVERSITY

29. October 2025

Magnus Bender  
Assistant Professor

34



- Making tools (self defined local Python functions) available to ChatGPT

# FUNCTION CALLING AND THE OPEN AI API: GETTING THE ANSWER

---

```
response = client.responses.create(  
    model="gpt-5-nano",  
    instructions="Respond only with a list  
    of employees that are available for  
    the meeting in question.",  
    tools=tools,  
    input=messages,  
    # may use structured outputs etc.  
)  
  
print(response.output_text)
```

Mary  
Elizabeth



DEPARTMENT OF MANAGEMENT  
AARHUS UNIVERSITY

29. October 2025

Magnus Bender  
Assistant Professor

34



- Making tools (self defined local Python functions) available to ChatGPT

1. Define functions and tell ChatGPT

# FUNCTION CALLING AND THE OPEN AI API: GETTING THE ANSWER

---

```
response = client.responses.create(  
    model="gpt-5-nano",  
    instructions="Respond only with a list  
    of employees that are available for  
    the meeting in question.",  
    tools=tools,  
    input=messages,  
    # may use structured outputs etc.  
)  
  
print(response.output_text)
```

Mary  
Elizabeth



DEPARTMENT OF MANAGEMENT  
AARHUS UNIVERSITY

29. October 2025

Magnus Bender  
Assistant Professor

34



- Making tools (self defined local Python functions) available to ChatGPT
  1. Define functions and tell ChatGPT
  2. Call OpenAI API with the question and the available tools
    - Returns the functions calls (function names & parameters) to which ChatGPT needs the return values

# FUNCTION CALLING AND THE OPEN AI API: GETTING THE ANSWER

---

```
response = client.responses.create(  
    model="gpt-5-nano",  
    instructions="Respond only with a list  
    of employees that are available for  
    the meeting in question.",  
    tools=tools,  
    input=messages,  
    # may use structured outputs etc.  
)  
  
print(response.output_text)
```

Mary  
Elizabeth



- Making tools (self defined local Python functions) available to ChatGPT
  1. Define functions and tell ChatGPT
  2. Call OpenAI API with the question and the available tools
    - Returns the functions calls (function names & parameters) to which ChatGPT needs the return values
  3. Run the functions calls locally in Python

# FUNCTION CALLING AND THE OPEN AI API: GETTING THE ANSWER

---

```
response = client.responses.create(  
    model="gpt-5-nano",  
    instructions="Respond only with a list  
    of employees that are available for  
    the meeting in question.",  
    tools=tools,  
    input=messages,  
    # may use structured outputs etc.  
)
```

```
print(response.output_text)
```

Mary  
Elizabeth

- Making tools (self defined local Python functions) available to ChatGPT
  1. Define functions and tell ChatGPT
  2. Call OpenAI API with the question and the available tools
    - Returns the functions calls (function names & parameters) to which ChatGPT needs the return values
  3. Run the functions calls locally in Python
  4. Call OpenAI API again, now with the return values of all function calls

# FUNCTION CALLING AND THE OPEN AI API: GETTING THE ANSWER

---

```
response = client.responses.create(  
    model="gpt-5-nano",  
    instructions="Respond only with a list  
    of employees that are available for  
    the meeting in question.",  
    tools=tools,  
    input=messages,  
    # may use structured outputs etc.  
)
```

```
print(response.output_text)
```

Mary  
Elizabeth

- Making tools (self defined local Python functions) available to ChatGPT
  1. Define functions and tell ChatGPT
  2. Call OpenAI API with the question and the available tools
    - Returns the functions calls (function names & parameters) to which ChatGPT needs the return values
  3. Run the functions calls locally in Python
  4. Call OpenAI API again, now with the return values of all function calls
  5. Get an overall answer

# FUNCTION CALLING AND THE OPEN AI API: GETTING THE ANSWER

---

```
response = client.responses.create(  
    model="gpt-5-nano",  
    instructions="Respond only with a list  
    of employees that are available for  
    the meeting in question.",  
    tools=tools,  
    input=messages,  
    # may use structured outputs etc.  
)
```

```
print(response.output_text)
```

Mary  
Elizabeth

Code available on  
uCloud!

- Making tools (self defined local Python functions) available to ChatGPT
  1. Define functions and tell ChatGPT
  2. Call OpenAI API with the question and the available tools
    - Returns the functions calls (function names & parameters) to which ChatGPT needs the return values
  3. Run the functions calls locally in Python
  4. Call OpenAI API again, now with the return values of all function calls
  5. Get an overall answer

# PYTHON PACKAGES

---

- Why we need them
- How to manage them

# PACKAGES & IMPORTS

---

- Some functions are always available  
`len()`, `str()`, `int()`, `range()`, `print()`

# PACKAGES & IMPORTS

---

- Some functions are always available  
`len()`, `str()`, `int()`, `range()`, `print()`
- Other functions need to be imported first  
`import time` → `time.time()`  
`import json` → `json.loads()`, `json.dumps()`

# PACKAGES & IMPORTS

---

- Some functions are always available  
`len()`, `str()`, `int()`, `range()`, `print()`
- Other functions need to be imported first  
`import time` → `time.time()`  
`import json` → `json.loads()`, `json.dumps()`
- The `import` keyword loads a so-called module and makes it available for use

# PACKAGES & IMPORTS

---

- Some functions are always available
  - len(), str(), int(), range(), print()
- Other functions need to be imported first
  - import time → time.time()
  - import json → json.loads(), json.dumps()
- The import keyword loads a so-called module and makes it available for use
  - Modules may be part of python (standard library), e.g., json, csv, time

# PACKAGES & IMPORTS

---

- Some functions are always available
  - len(), str(), int(), range(), print()
- Other functions need to be imported first
  - import time → time.time()
  - import json → json.loads(), json.dumps()
- The import keyword loads a so-called module and makes it available for use
  - Modules may be part of python (standard library), e.g., json, csv, time
  - Modules may be part of packages, e.g., pydantic, nicegui, openai

# PACKAGES & IMPORTS

---

- Some functions are always available
  - len(), str(), int(), range(), print()
- Other functions need to be imported first
  - import time → time.time()
  - import json → json.loads(), json.dumps()
- The import keyword loads a so-called module and makes it available for use
  - Modules may be part of python (standard library), e.g., json, csv, time
  - Modules may be part of packages, e.g., pydantic, nicegui, openai
  - Modules may be self-created, custom modules

# IMPORT CUSTOM MODULES

---

example\_import.py

```
def hello():
    print("Hello World!")

def bye():
    print("Bye!")

VARIABLE = False
```

# IMPORT CUSTOM MODULES

---

name.py

```
import example_import

print(example_import)
print(example_import.bye, example_import.hello)

print(example_import.VARIABLE)
example_import.bye()
example_import.hello()
```

example\_import.py

```
def hello():
    print("Hello World!")

def bye():
    print("Bye!")

VARIABLE = False
```

- Create two files: example\_import.py and name.py

# IMPORT CUSTOM MODULES

---

name.py

```
import example_import

print(example_import)
print(example_import.bye, example_import.hello)

print(example_import.VARIABLE)
example_import.bye()
example_import.hello()
```

example\_import.py

```
def hello():
    print("Hello World!")

def bye():
    print("Bye!")

VARIABLE = False
```

- Create two files: `example_import.py` and `name.py`
- Import `example_import` in `name.py`

# IMPORT CUSTOM MODULES

---

name.py

```
import example_import

print(example_import)
print(example_import.bye, example_import.hello)

print(example_import.VARIABLE)
example_import.bye()
example_import.hello()
```

```
<module 'example_import' from './example_import.py'>
<function bye at 0x100be7760>
    <function hello at 0x100be6200>
```

example\_import.py

```
def hello():
    print("Hello World!")

def bye():
    print("Bye!")

VARIABLE = False
```

- Create two files: example\_import.py and name.py
- Import example\_import in name.py
  - example\_import becomes a module

# IMPORT CUSTOM MODULES

---

name.py

```
import example_import

print(example_import)
print(example_import.bye, example_import.hello)

print(example_import.VARIABLE)
example_import.bye()
example_import.hello()

<module 'example_import' from './example_import.py'>
<function bye at 0x100be7760>
    <function hello at 0x100be6200>
```

example\_import.py

```
def hello():
    print("Hello World!")

def bye():
    print("Bye!")

VARIABLE = False
```

- Create two files: `example_import.py` and `name.py`
- Import `example_import` in `name.py`
  - `example_import` becomes a module
  - Variables, classes, and functions in `example_import` are accessible via `example_import.<variable_name>`, `example_import.<class name>`, and `example_import.<function_name>`

# IMPORT CUSTOM MODULES

---

name.py

```
import example_import

print(example_import)
print(example_import.bye, example_import.hello)

print(example_import.VARIABLE)
example_import.bye()
example_import.hello()
```

```
<module 'example_import' from './example_import.py'>
<function bye at 0x100be7760>
    <function hello at 0x100be6200>
```

```
False
Bye!
Hello World!
```



DEPARTMENT OF MANAGEMENT  
AARHUS UNIVERSITY

29. October 2025

Magnus Bender  
Assistant Professor

example\_import.py

```
def hello():
    print("Hello World!")

def bye():
    print("Bye!")

VARIABLE = False
```

- Create two files: `example_import.py` and `name.py`
- Import `example_import` in `name.py`
  - `example_import` becomes a module
  - Variables, classes, and functions in `example_import` are accessible via `example_import.<variable_name>`, `example_import.<class name>`, and `example_import.<function_name>`

# MORE IMPORT STATEMENTS

---

name.py

```
import example_import
from example_import import hello

print(example_import.hello)
print(hello)
hello()
```

```
<function hello at 0x100be6200>
<function hello at 0x100be6200>
Hello World!
```

```
import sys, time

from example_import import *

import example_import as ei
```

example\_import.py

```
def hello():
    print("Hello World!")

def bye():
    print("Bye!")

VARIABLE = False
```

# MORE IMPORT STATEMENTS

name.py

```
import example_import
from example_import import hello

print(example_import.hello)
print(hello)
hello()
```

Import a name from a module

example\_import.py

```
def hello():
    print("Hello World!")

def bye():
    print("Bye!")

VARIABLE = False
```

```
import sys, time

from example_import import *

import example_import as ei
```

# MORE IMPORT STATEMENTS

---

name.py

```
import example_import
from example_import import hello

print(example_import.hello)
print(hello)
hello()
```

```
import sys, time

from example_import import *

import example_import as ei
```

example\_import.py

```
def hello():
    print("Hello World!")

def bye():
    print("Bye!")

VARIABLE = False
```

The same function, but two ways to access it.

```
<function hello at 0x100be6200>
<function hello at 0x100be6200>
Hello World!
```

# MORE IMPORT STATEMENTS

---

name.py

```
import example_import
from example_import import hello

print(example_import.hello)
print(hello)
hello()
```

```
<function hello at 0x100be6200>
<function hello at 0x100be6200>
Hello World!
```

```
import sys, time
from example_import import *
import example_import as ei
```

Import two modules with one import statement

example\_import.py

```
def hello():
    print("Hello World!")

def bye():
    print("Bye!")

VARIABLE = False
```

# MORE IMPORT STATEMENTS

---

name.py

```
import example_import
from example_import import hello

print(example_import.hello)
print(hello)
hello()
```

```
<function hello at 0x100be6200>
<function hello at 0x100be6200>
Hello World!
```

```
import sys, time

from example_import import *
import example_import as ei
```

Import all variables, classes, and functions from a module

example\_import.py

```
def hello():
    print("Hello World!")

def bye():
    print("Bye!")

VARIABLE = False
```

# MORE IMPORT STATEMENTS

---

name.py

```
import example_import
from example_import import hello

print(example_import.hello)
print(hello)
hello()
```

```
<function hello at 0x100be6200>
<function hello at 0x100be6200>
Hello World!
```

```
import sys, time

from example_import import *

import example_import as ei
```

Import with a different name, use ei.<name>

example\_import.py

```
def hello():
    print("Hello World!")

def bye():
    print("Bye!")

VARIABLE = False
```

# HOW TO GET PACKAGES & MODULES

---

# HOW TO GET PACKAGES & MODULES

---

- Modules may be part of Python (standard library), e.g., `json`, `csv`, `time`
  - No need to install, are already available

# HOW TO GET PACKAGES & MODULES

---

- Modules may be part of Python (standard library), e.g., `json`, `csv`, `time`
  - No need to install, are already available
- Modules may be part of packages, e.g., `pydantic`, `nicegui`, `openai`

# HOW TO GET PACKAGES & MODULES

---

- Modules may be part of Python (standard library), e.g., `json`, `csv`, `time`
  - No need to install, are already available
- Modules may be part of packages, e.g., `pydantic`, `nicegui`, `openai`
  - Need to be installed using `pip` (Python package installer)
  - Installs packages from PyPi (<https://pypi.org/>)

# HOW TO GET PACKAGES & MODULES

---

- Modules may be part of Python (standard library), e.g., `json`, `csv`, `time`
  - No need to install, are already available
- Modules may be part of packages, e.g., `pydantic`, `nicegui`, `openai`
  - Need to be installed using `pip` (Python package installer)
  - Installs packages from PyPi (<https://pypi.org/>)
    - Possible to search there for relevant packages
    - Possible to ask ChatGPT
    - We introduced you to some useful packages

# HOW TO GET PACKAGES & MODULES

---

- Modules may be part of Python (standard library), e.g., `json`, `csv`, `time`
  - No need to install, are already available
- Modules may be part of packages, e.g., `pydantic`, `nicegui`, `openai`
  - Need to be installed using `pip` (Python package installer)
  - Installs packages from PyPi (<https://pypi.org/>)
    - Possible to search there for relevant packages
    - Possible to ask ChatGPT
    - We introduced you to some useful packages
- Modules may be self-created, custom modules
  - Easy way: Place multiple Python files in same directory and use `import <file name>`

# PIP: PYTHON PACKAGE INSTALLER

---

# PIP: PYTHON PACKAGE INSTALLER

---

- Each of us already uses pip on uCloud

# PIP: PYTHON PACKAGE INSTALLER

---

- Each of us already uses pip on uCloud
  - But we did not tell you about (only select „initialization setup.sh“)

# PIP: PYTHON PACKAGE INSTALLER

---

- Each of us already uses pip on uCloud
  - But we did not tell you about (only select „initialization setup.sh“)
- On uCloud:

# PIP: PYTHON PACKAGE INSTALLER

---

- Each of us already uses pip on uCloud
  - But we did not tell you about (only select „initialization setup.sh“)
- On uCloud:
  - Each of you has your own Python environment

# PIP: PYTHON PACKAGE INSTALLER

---

- Each of us already uses pip on uCloud
  - But we did not tell you about (only select „initialization setup.sh“)
- On uCloud:
  - Each of you has your own Python environment
  - All modules from the standard library
  - Some additional packages, pre-installed by us via pip

# PIP: PYTHON PACKAGE INSTALLER

---

- Each of us already uses pip on uCloud
  - But we did not tell you about (only select „initialization setup.sh“)
- On uCloud:
  - Each of you has your own Python environment
  - All modules from the standard library
  - Some additional packages, pre-installed by us via pip
    - ▶ See the `requirements.txt`



`requirements.txt`

requests  
tqdm  
pdoc

pydantic  
nicegui

nltk  
numpy  
scipy  
scikit-learn

openai  
ipykernel

pdfminer.six

# PIP: PYTHON PACKAGE INSTALLER

---

- Each of us already uses pip on uCloud
  - But we did not tell you about (only select „initialization setup.sh“)
- On uCloud:
  - Each of you has your own Python environment
  - All modules from the standard library
  - Some additional packages, pre-installed by us via pip
    - ▶ See the `requirements.txt` →
  - You may install more packages if you need them

`requirements.txt`

`requests`  
`tqdm`  
`pdoc`

`pydantic`  
`nicegui`

`nltk`  
`numpy`  
`scipy`  
`scikit-learn`

`openai`  
`ipykernel`

`pdfminer.six`

# PIP: PYTHON PACKAGE INSTALLER

---

- Each of us already uses pip on uCloud
  - But we did not tell you about (only select „initialization setup.sh“)
- On uCloud:
  - Each of you has your own Python environment
  - All modules from the standard library
  - Some additional packages, pre-installed by us via pip
    - ▶ See the `requirements.txt` →
  - You may install more packages if you need them
    - ▶ `pip list`
    - ▶ `pip install <package name>`

`requirements.txt`

`requests`  
`tqdm`  
`pdoc`

`pydantic`  
`nicegui`

`nltk`  
`numpy`  
`scipy`  
`scikit-learn`

`openai`  
`ipykernel`

`pdfminer.six`

# PIP: PYTHON PACKAGE INSTALLER

---

- Each of us already uses pip on uCloud
  - But we did not tell you about (only select „initialization setup.sh“)
- On uCloud:
  - Each of you has your own Python environment
  - All modules from the standard library
  - Some additional packages, pre-installed by us via pip
    - ▶ See the `requirements.txt` →
  - You may install more packages if you need them
    - ▶ `pip list`
    - ▶ `pip install <package name>`
- We will do it together in the next tutorial!

`requirements.txt`

`requests`  
`tqdm`  
`pdoc`

`pydantic`  
`nicegui`

`nltk`  
`numpy`  
`scipy`  
`scikit-learn`

`openai`  
`ipykernel`

`pdfminer.six`

# SUMMARIZE TODAY

---

- Take home messages.

# GET A FEELING FOR RAG

---

- How to identify good parameters?

# GET A FEELING FOR RAG

---

- How to identify good parameters?
  - Chunk size:
    - Do we actually need a fixed size?

# GET A FEELING FOR RAG

---

- How to identify good parameters?
  - Chunk size:
    - Do we actually need a fixed size?
      - Or would it be better use the text structure, i.e., the pages, sections, and paragraphs of documents.
      - Can we think of a way to dynamically determine the chunk size?

# GET A FEELING FOR RAG

---

- How to identify good parameters?
  - Chunk size:
    - Do we actually need a fixed size?
      - Or would it be better use the text structure, i.e., the pages, sections, and paragraphs of documents.
      - Can we think of a way to dynamically determine the chunk size?
    - A chunk must be large enough to contain full content, but not too large.
    - Should overlapping be used to circumvent cut contexts and if yes, how much overlap?

# GET A FEELING FOR RAG

---

- How to identify good parameters?
  - Chunk size:
    - Do we actually need a fixed size?
      - Or would it be better use the text structure, i.e., the pages, sections, and paragraphs of documents.
      - Can we think of a way to dynamically determine the chunk size?
    - A chunk must be large enough to contain full content, but not too large.
    - Should overlapping be used to circumvent cut contexts and if yes, how much overlap?
  - Why, when, and how should we augment questions (before doing the retrieval)?

# GET A FEELING FOR RAG

---

- How to identify good parameters?
  - Chunk size:
    - Do we actually need a fixed size?
      - Or would it be better use the text structure, i.e., the pages, sections, and paragraphs of documents.
      - Can we think of a way to dynamically determine the chunk size?
    - A chunk must be large enough to contain full content, but not too large.
    - Should overlapping be used to circumvent cut contexts and if yes, how much overlap?
  - Why, when, and how should we augment questions (before doing the retrieval)?
  - How many items to add during augmentation?

# GET A FEELING FOR RAG

---

- How to identify good parameters?
  - Chunk size:
    - Do we actually need a fixed size?
      - Or would it be better use the text structure, i.e., the pages, sections, and paragraphs of documents.
      - Can we think of a way to dynamically determine the chunk size?
    - A chunk must be large enough to contain full content, but not too large.
    - Should overlapping be used to circumvent cut contexts and if yes, how much overlap?
  - Why, when, and how should we augment questions (before doing the retrieval)?
  - How many items to add during augmentation?
    - Use a fixed threshold for similarity, a fixed number of items?
    - How to dynamically gather good values?

# TUTORIAL ON FRIDAY AND MONDAY

---

# TUTORIAL ON FRIDAY AND MONDAY

---

## 1. Use pip

- Get an overview of installed packages
- Install more packages
- Get a list of packages to install the same selection elsewhere

# TUTORIAL ON FRIDAY AND MONDAY

---

1. Use pip
  - Get an overview of installed packages
  - Install more packages
  - Get a list of packages to install the same selection elsewhere
2. Implement a RAG-pipeline on your own
  - There will be a partly implemented pipeline in a notebook
  - There will be a GUI using your pipeline in the notebook

# TUTORIAL ON FRIDAY AND MONDAY

---

1. Use pip
  - Get an overview of installed packages
  - Install more packages
  - Get a list of packages to install the same selection elsewhere
2. Implement a RAG-pipeline on your own
  - There will be a partly implemented pipeline in a notebook
  - There will be a GUI using your pipeline in the notebook
3. Use RAG for your project (*optional*)

# TL;DRL

---

- RAG



DEPARTMENT OF MANAGEMENT  
AARHUS UNIVERSITY

29. October 2025

Magnus Bender  
Assistant Professor

44



# TL;DRL

---

- RAG
  - RAG is actually what happens if you, e.g., upload files to ChatGPT or Google NotebookLM and ask questions about the content.

# TL;DRL

---

- RAG
  - RAG is actually what happens if you, e.g., upload files to ChatGPT or Google NotebookLM and ask questions about the content.
  - Allows us to avoid training specific models

# TL;DRL

---

- RAG
  - RAG is actually what happens if you, e.g., upload files to ChatGPT or Google NotebookLM and ask questions about the content.
  - Allows us to avoid training specific models
  - Follows the idea to add relevant context (context from a dataset) to each prompt, the relevant context is retrieved using embeddings and their similarity to the question/prompt

# TL;DRL

---

- RAG
  - RAG is actually what happens if you, e.g., upload files to ChatGPT or Google NotebookLM and ask questions about the content.
  - Allows us to avoid training specific models
  - Follows the idea to add relevant context (context from a dataset) to each prompt, the relevant context is retrieved using embeddings and their similarity to the question/prompt
- Function calling with LLMs
  - Make tools (Python functions) available to ChatGPT, let it choose which function to run and then take the function's return values into account

# TL;DRL

---

- RAG
  - RAG is actually what happens if you, e.g., upload files to ChatGPT or Google NotebookLM and ask questions about the content.
  - Allows us to avoid training specific models
  - Follows the idea to add relevant context (context from a dataset) to each prompt, the relevant context is retrieved using embeddings and their similarity to the question/prompt
- Function calling with LLMs
  - Make tools (Python functions) available to ChatGPT, let it choose which function to run and then take the function's return values into account
- Python packages
  - Adding more functionality by installing external code



**DEPARTMENT OF MANAGEMENT**  
AARHUS UNIVERSITY