COLUMBIA+

Module 2

Programming an LLM

Hardeep Johar

Senior Lecturer in the Discipline of
Industrial Engineering and Operations Research

LLM Programming Tools

LLMs come with support for programmers through platforms, APIs, and third party libraries

OpenAI API

Meta LLaMa 3

Google Gemini
Vertex API

What is an API?

An API allows you to:

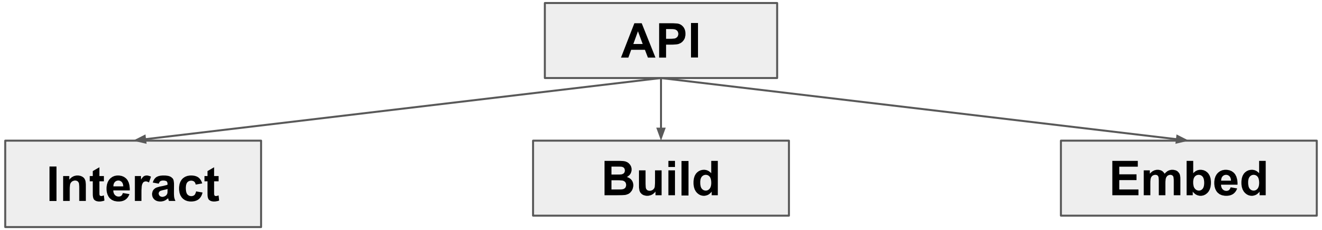
- ❏ Write complex threaded queries
- ❏ Extract relevant responses
- ❏ Build code
- ❏ Tailor an LLM to your specific domain
- ❏ Embed the LLM in your own applications

OpenAI API

The OpenAI API allows you to programmatically access various GPT models

OpenAI API Docs: https://platform.openai.com/docs/overview

OpenAI API



Interact with a GPT LLM in the same manner as ChatGPT but from inside a program

- Ask questions
- Chat with the model
- Analyze data (input data into the LLM)
- Get programming suggestions or fix code

Build a specialized version of the LLM for your own application or organization

Embed a GPT based chatbot in your own web or mobile application - seamless integration of the LLM into a broader application

OpenAI API: What You Need

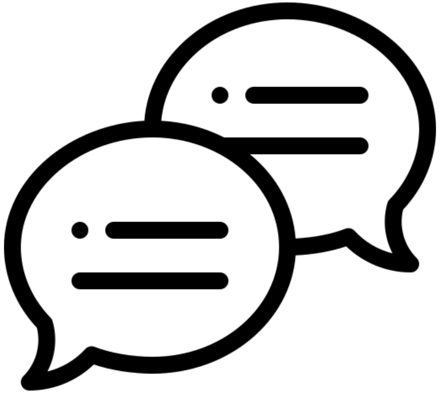
- An OpenAI account
  - https://platform.openai.com/signup
- Create a project
  - https://platform.openai.com/organization/projects
- Create a secret API key
  - https://platform.openai.com/api-keys
- Store the key in a safe place
  - The “Colab notebooks” folder on your Google Drive is probably the easiest
  - Don’t share it with anyone!

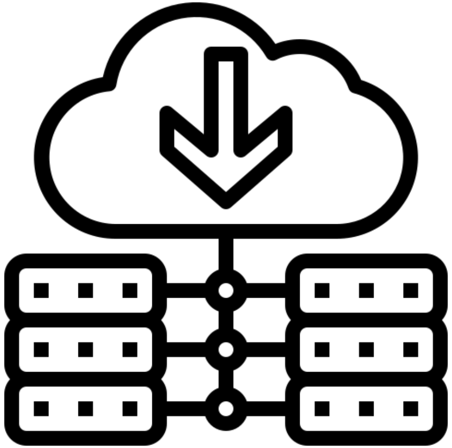
Summary

LLM Programming Tools
What is an API?
OpenAI API

OpenAI API: Using the Key

OpenAI API Notebook





Flight number: 42

Arrival time: 4:00 pm

Flight number: 42

Arrival time: 4:10pm

Examples of applications

- ❑ Customer service chatbots
- ❑ Discovering new proteins
- ❑ Symptom diagnosis
- ❑ Accessibility
- ❑ Learning history
- ❑ Document summarization
- ❑ Search engines
- ❑ Video editing
- ❑ Marketing campaigns
- ❑ Accounting

Building Custom LLMs

Human Agent vs LLM Agent - 1

A human agent

- Has some basic knowledge of the world (e.g., a college degree)
- Can converse in natural language
- Can answer general questions
- Can make deductive inferences
- But doesn’t know the internal business knowledge to answer business specific questions

An LLM agent

- Has been trained on large amounts of text data
- Can converse in natural language
- Can answer general questions
- Has some ability to make deductions as long as they have been trained on relevant information
- But doesn’t know the internal business knowledge to answer business specific questions

Human Agent vs LLM Agent - 2

- A human agent can add to knowledge

Human

By going back to school

- a graduate degree, professional certifications, etc.
- Usually requires a significant financial outlay

By being provided in-house training

- an onboarding program at an organization
- Trade subscriptions
- Usually a smaller cost

Human Agent vs LLM Agent - 3

- An LLM agent can add to knowledge

LLMs

By Being Retrained

Very expensive
(analogous to sending a human agent back to elementary school!)

Fine Tuning

- Adjusting model parameters using supervised learning
- **Expensive and time consuming** (analogous to sending a human agent to graduate school or to a professional certification program)

Retrieval Augmented Generation (RAG)

- Provide the model with specialized information and a mechanism to retrieve this information
- The model will use this specialized information and fall back to generalized knowledge if necessary
- Model parameters don’t change
- **Relatively inexpensive**

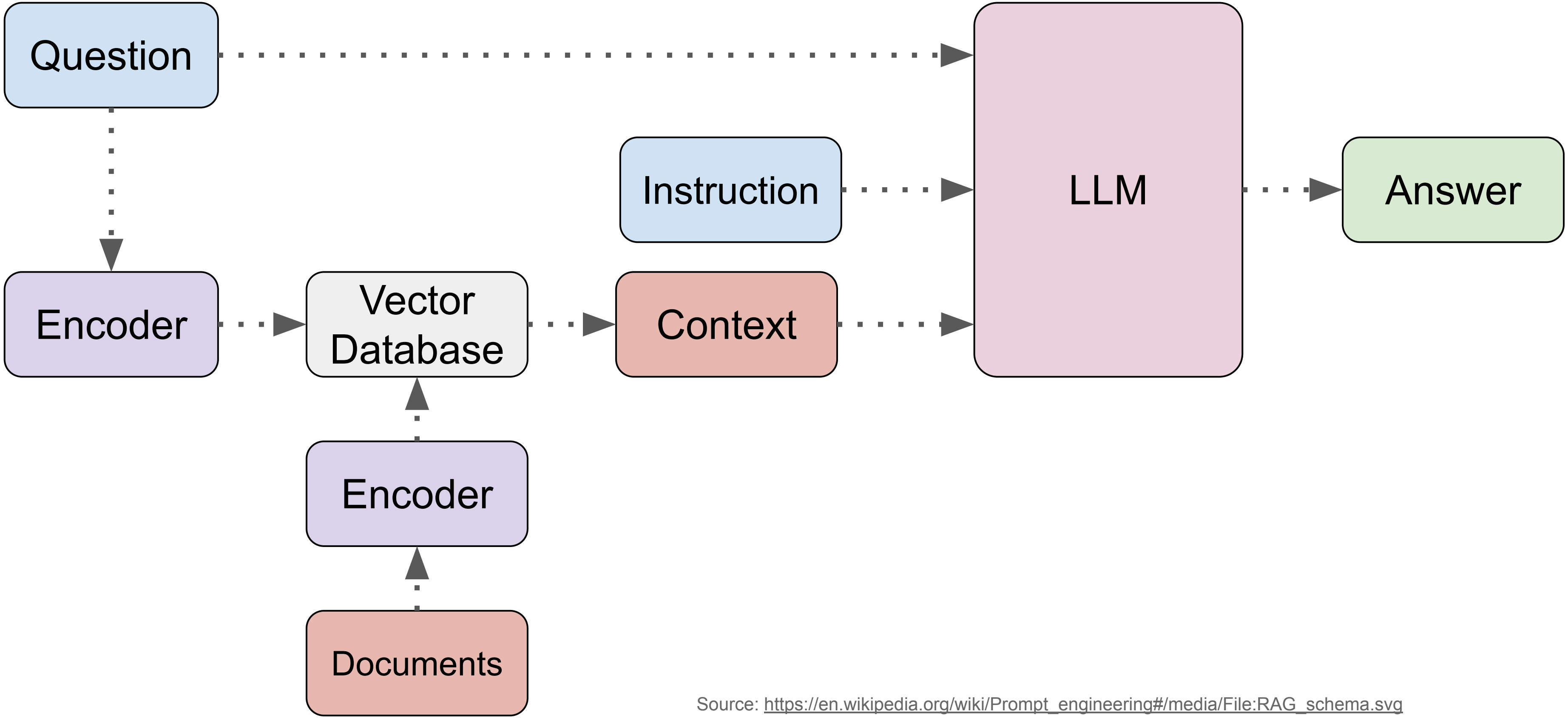
Retrieval-Augmented Generation - 1

- The LLM accesses information from an external (to its model) document repository
- It uses this document repository, as well as its trained model, to answer queries
- Roughly:
    - The documents are converted into chunks (short sequences of words)
    - The chunks are converted into embedded vectors
    - The query is converted into embedded vectors
    - The most similar document embedded vectors are chosen using a similarity algorithm
    - The LLM then uses its “language skills” to respond to the query

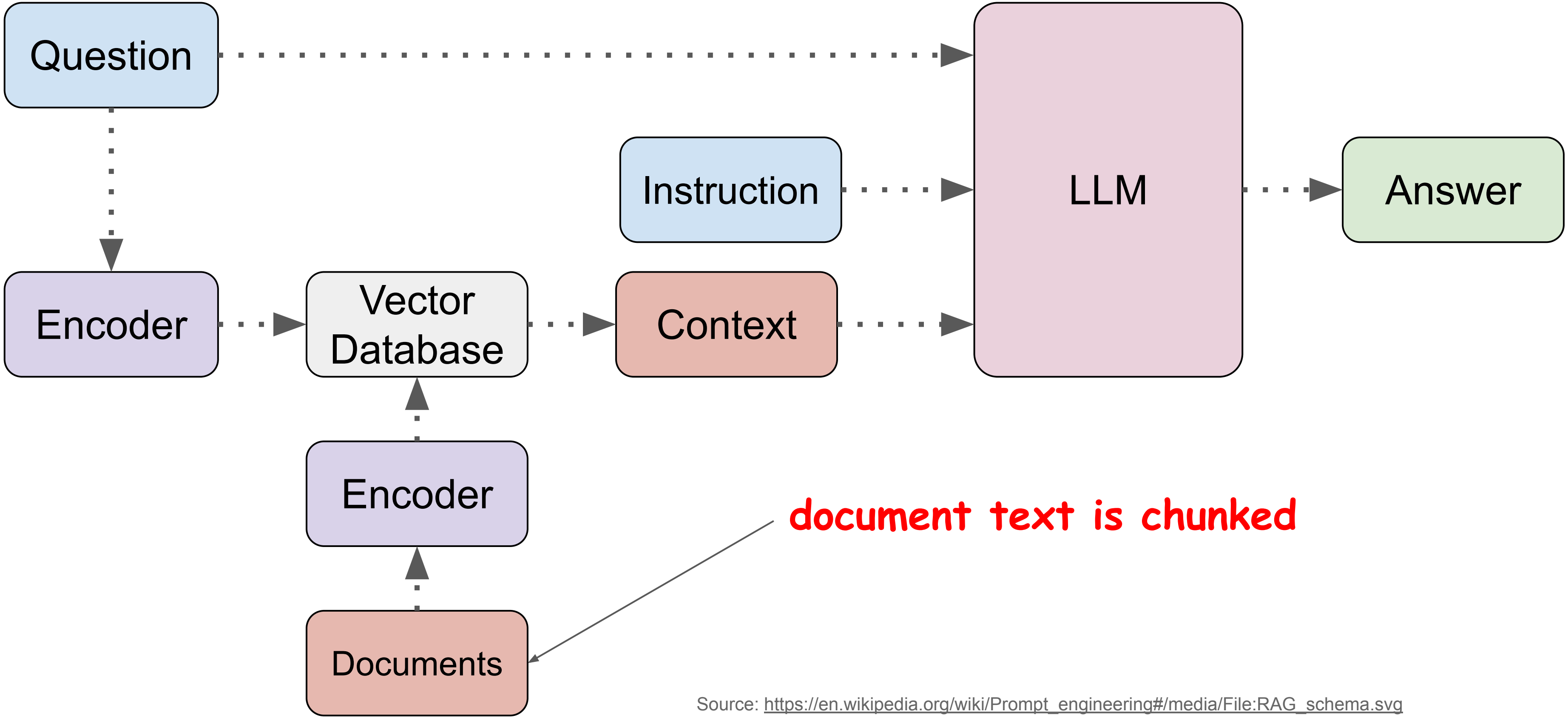
Embedding Chunked Vectors

- Since chunks are smaller groups of information
  - For example, 250 word chunks from a 10,000 word document
  - The likelihood of the “objects” in the chunks being related is high
  - Embedded vectors from these chunks will capture relationships better

Retrieval-Augmented Generation - 2



Retrieval-Augmented Generation - 3

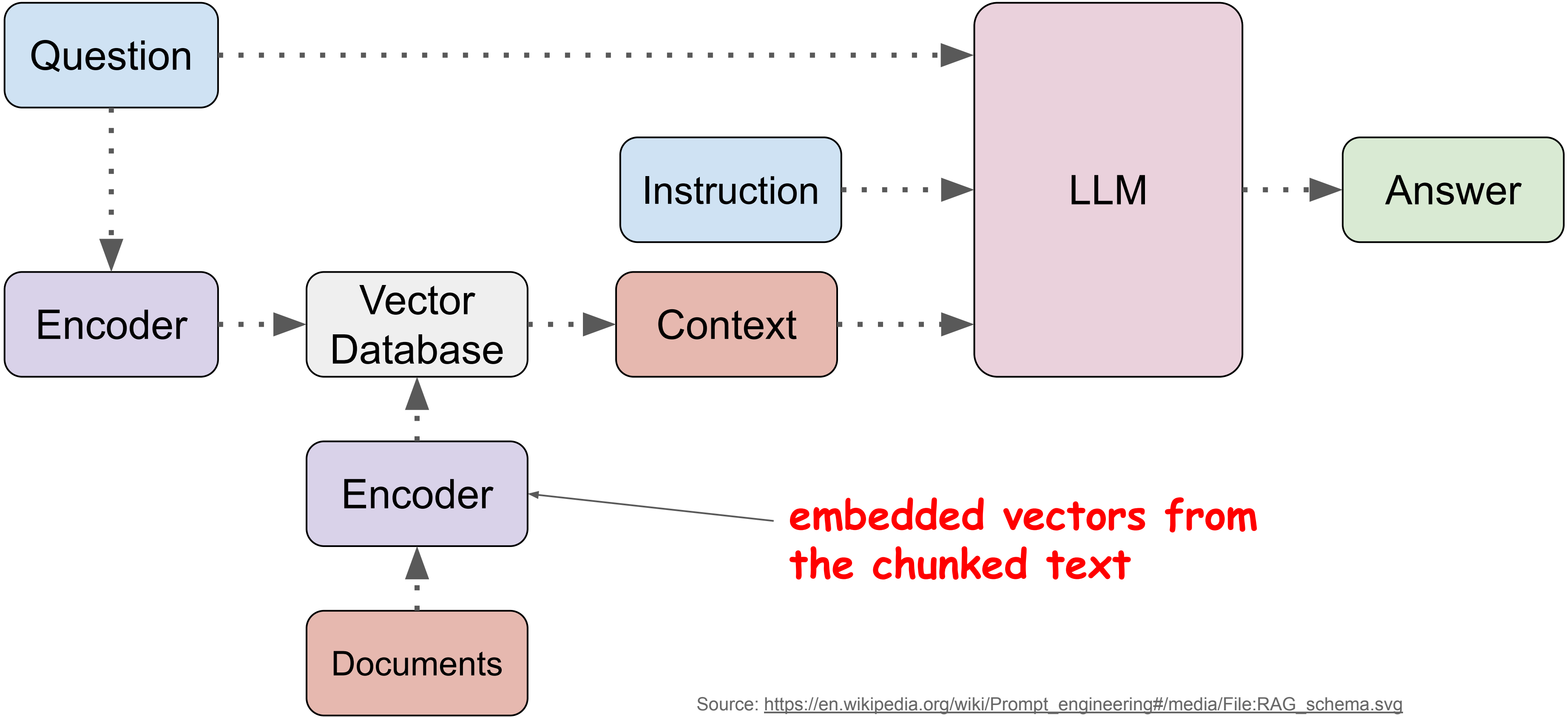


Source: https://en.wikipedia.org/wiki/Prompt_engineering#/media/File:RAG_schema.svg
By Gknor - Own work, CC BY-SA 4.0, https://commons.wikimedia.org/w/index.php?curid=139459823

Chunking

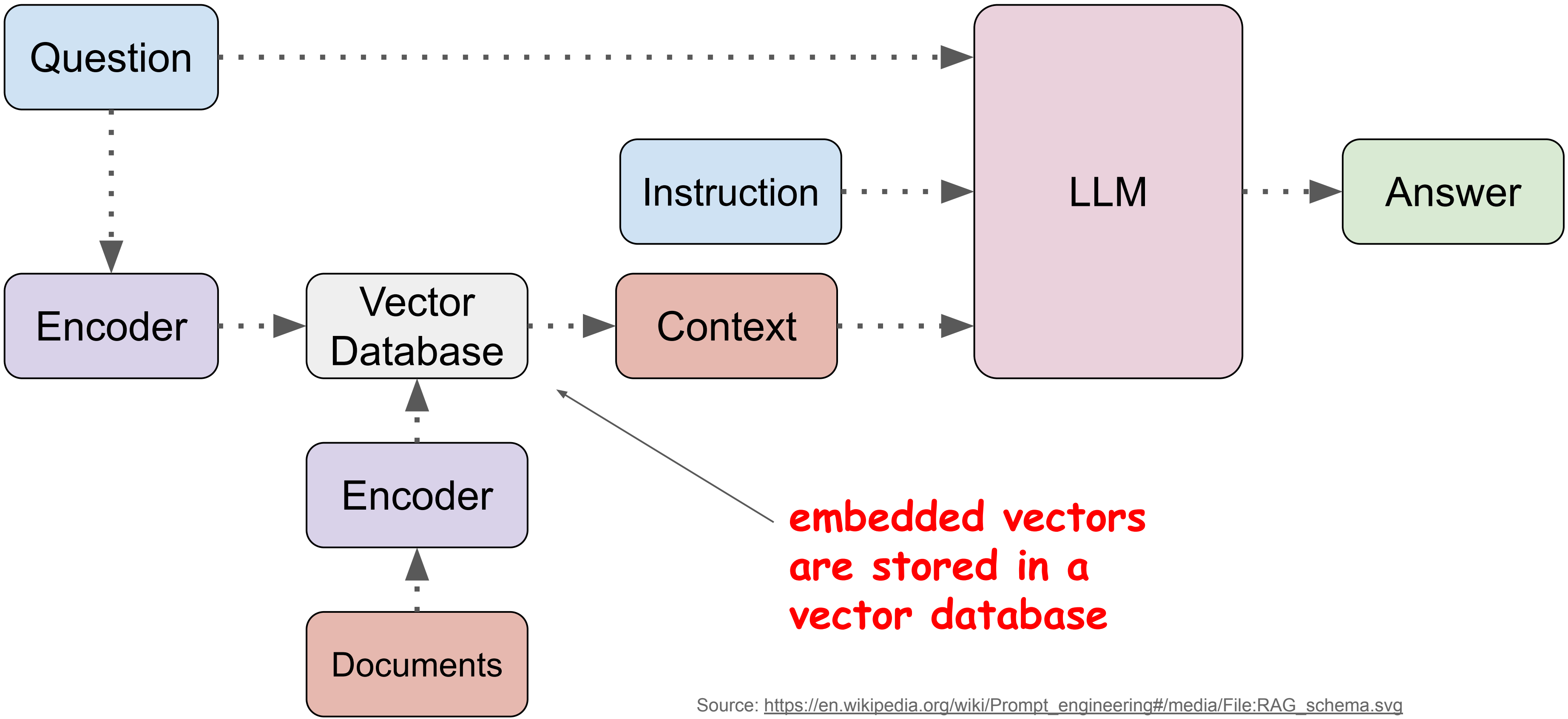
- Text documents are large and contain a variety of information
    - A customer service agent may contain information on
        - How to return products
            - With different policies for different products
        - How to contact a human agent
        - Locations of retail stores
- Chunking increases the probability that related information is grouped

Retrieval-Augmented Generation - 4



Source: https://en.wikipedia.org/wiki/Prompt_engineering#/media/File:RAG_schema.svg
By Gknor - Own work, CC BY-SA 4.0, https://commons.wikimedia.org/w/index.php?curid=139459823

Retrieval-Augmented Generation - 5



Source: https://en.wikipedia.org/wiki/Prompt_engineering#/media/File:RAG_schema.svg

Vector Databases and Vector Search

- Indexed databases for storing vectors
- Given a vector as input, vector databases search for matching vectors using a similarity algorithm
  - Cosine similarity:
    - Calculates the cosine of the angle between two vectors
    - The smaller the angle, the closer the cosine is to 1, and the more similar the vectors
- The number of vectors is large and calculating the similarity between every pair of vectors is computationally expensive
  - The input prompt is converted into a large number of vectors
  - The document repository is converted into a large number of vectors
  - The product of the two is large

Embedding Vectors and Cosine Similarity Notebook

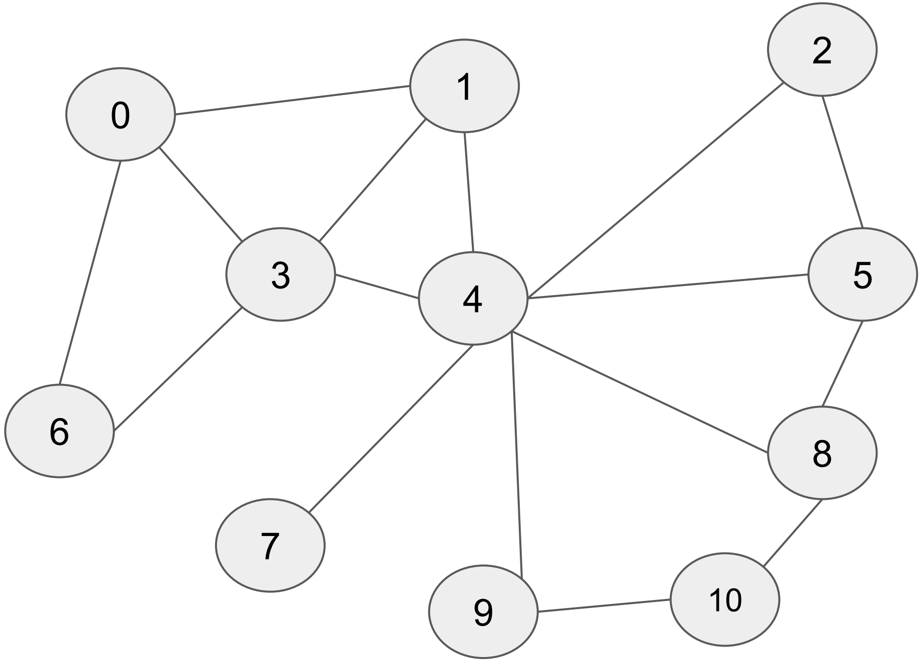
Vector Search: Retrieving Similar Vectors

Navigable Small Worlds (NSW) algorithm

- Store a pre-constructed similarity graph of document chunks
- Randomly pick a chunk and compute the similarity with the input vector
- Move to a neighbor of the chunk and recompute the similarity
- Stop when the similarity doesn’t get better
- Repeat and report the top-n similar chunks
- NSW algorithms rely on the “six degrees of separation” idea
    - The best similarity will be utmost some small n away from a random chunk

Navigable Small Worlds - 1

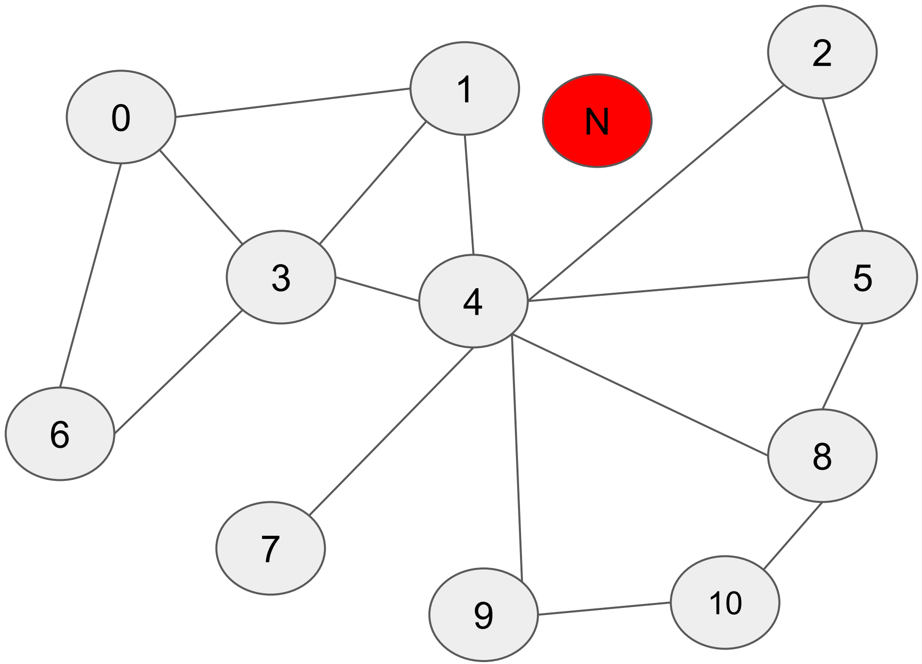
M=2



- Construct the base graph
- The parameter M specifies the number of connections a chunk makes to other chunks
- For a large number of chunks, the graph will be sparse
- The edge attribute is the inverse of cosine similarity

Navigable Small Worlds - 2

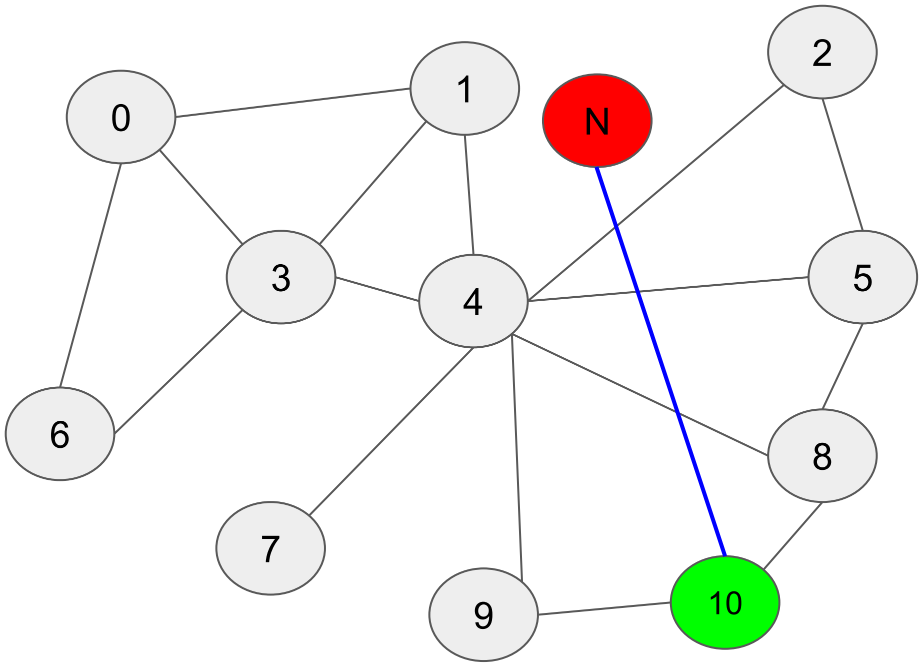
M=2



- A new chunk (from the LLM prompt) arrives
- And is inserted into the graph

Navigable Small Worlds - 3

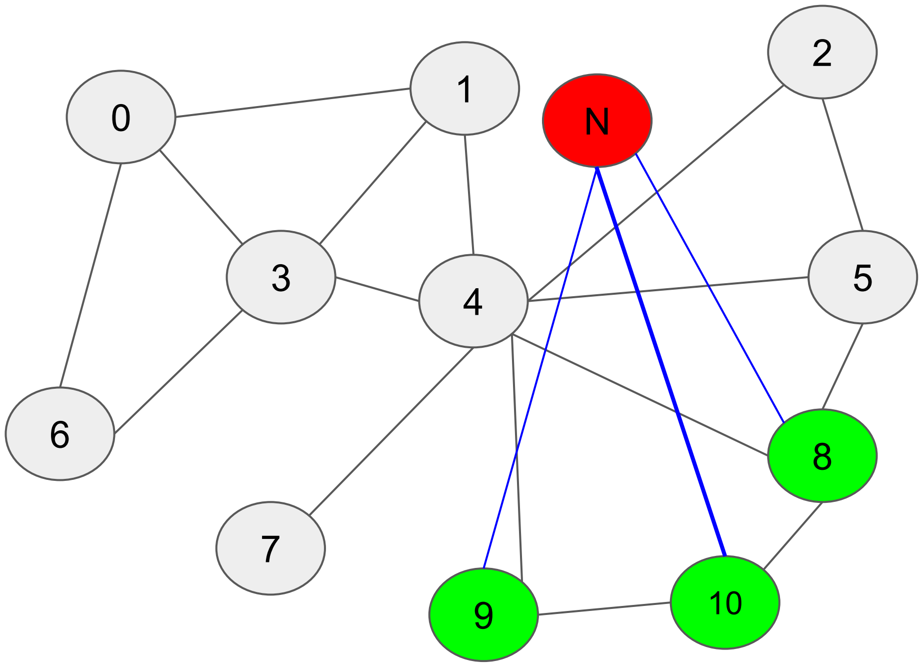
M=2



- Choose a random chunk (e.g., 10)
- And calculate the distance
    - add a new edge between 10 and N

Navigable Small Worlds - 4

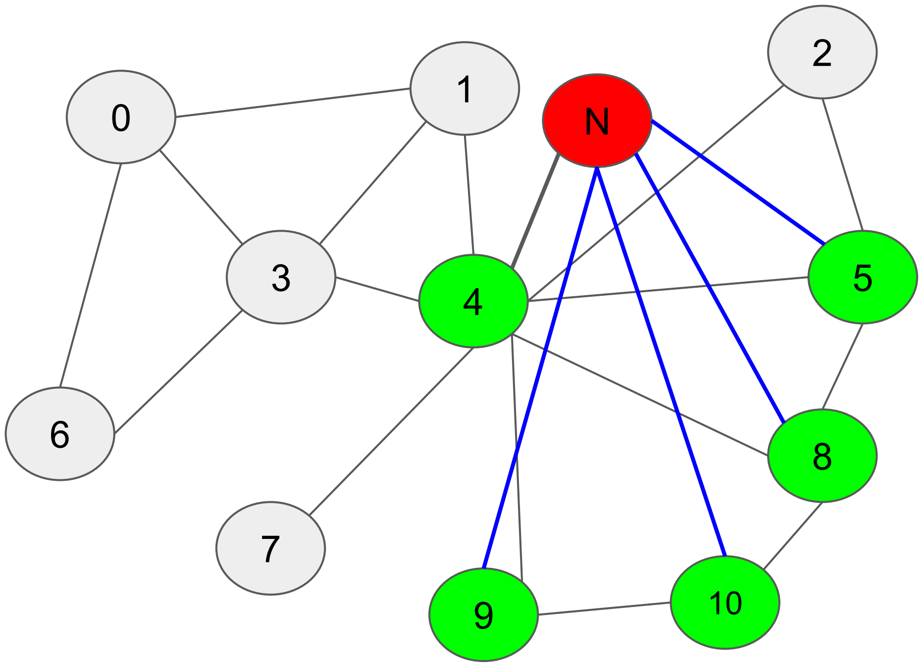
M=2



- Check distance (similarity) from two neighbors
- And calculate the distance
    - add new edges from 9 and 8 to N

Navigable Small Worlds - 5

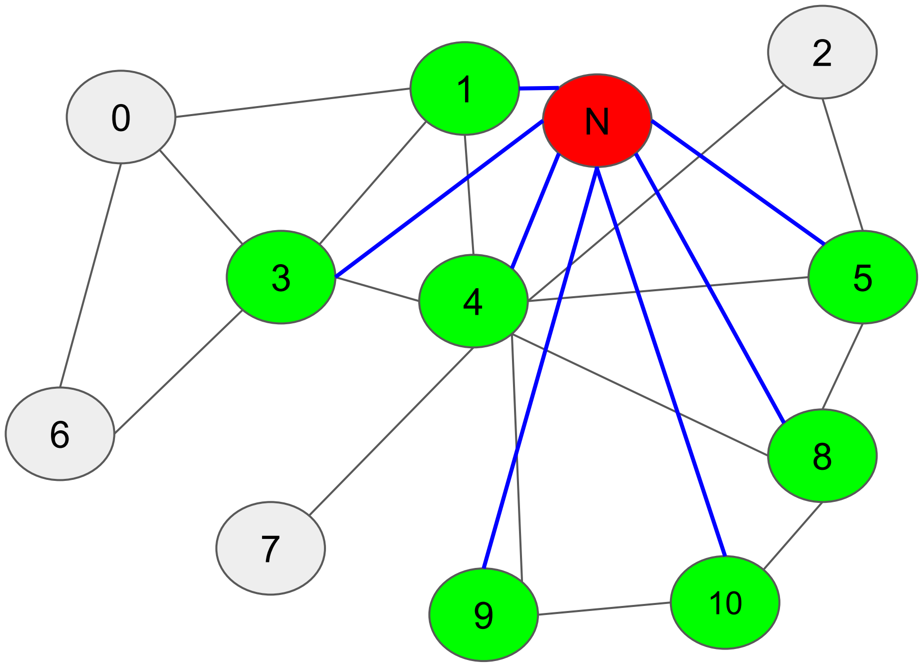
M=2



- 8 is closer
- Calculate distance from 4 and 5 to N

Navigable Small Worlds - 6

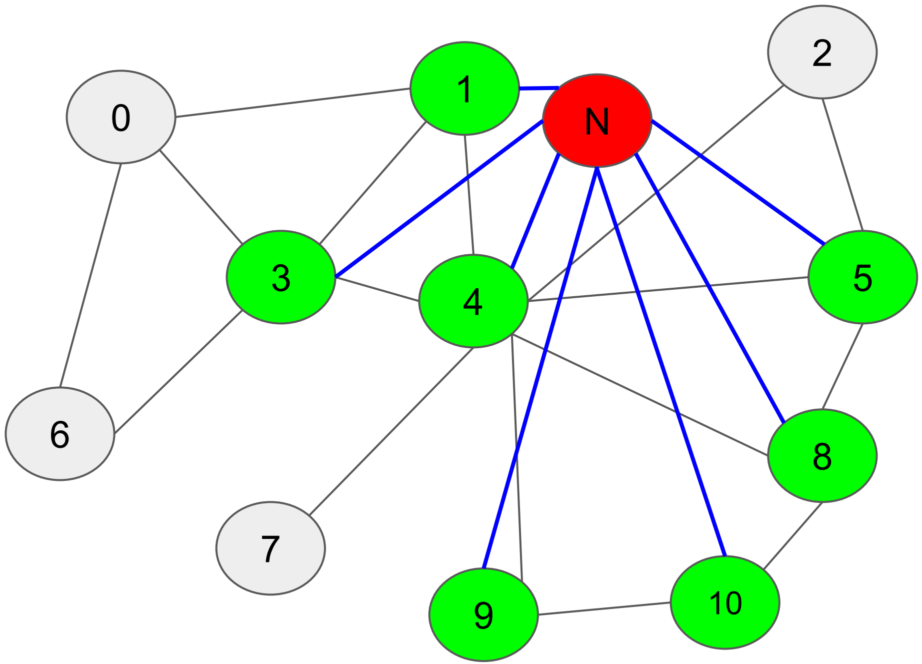
M=2



- 4 is closer
- Calculate distance from 1 and 3 to N

Navigable Small Worlds - 7

M=2

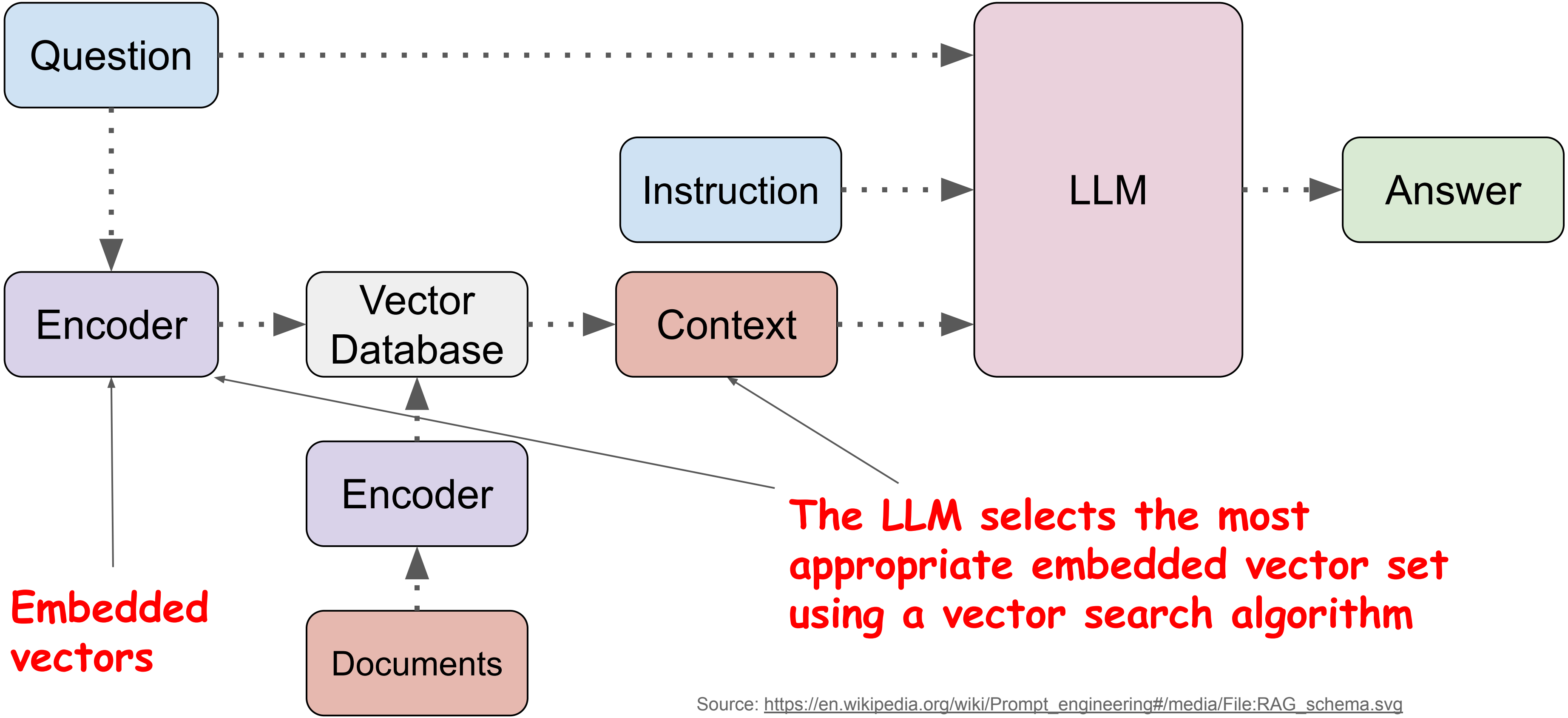


- 1 is closer
- Calculate distance from 0 and 3 to N
- Since neither 0 nor 3 is closer than 1, stop. 1 is the closest chunk

Vector Search: Retrieving Vectors

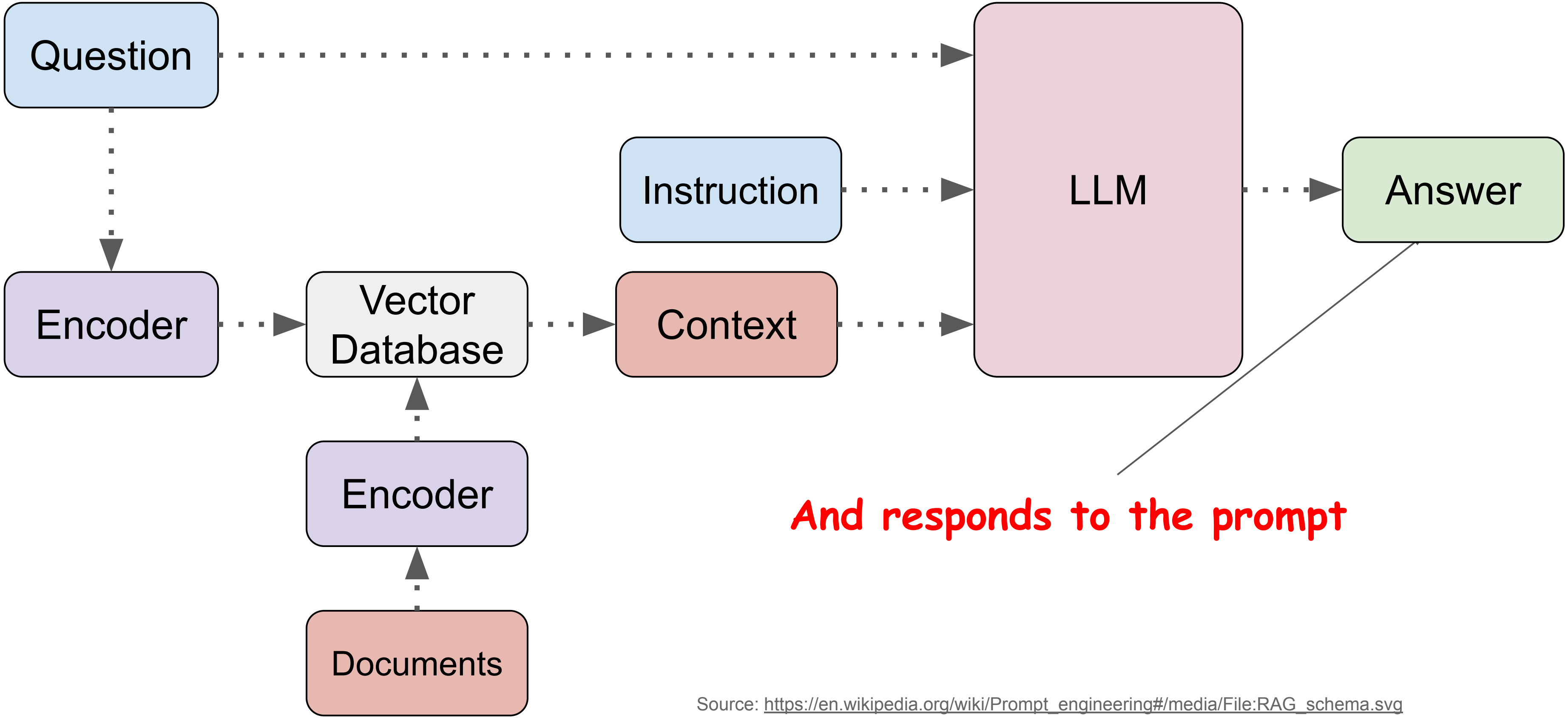
- Hierarchical Navigable Small Worlds (HNSW) algorithm
  - Adaptation of NSW but the pre-constructed graph is hierarchical with a small number of starting chunks with subsequent chunks arranged in a hierarchy
  - The algorithm picks a random chunk from the top level and then searches in that hierarchy
  - Using the NSW algorithm at each level
  - Facebook AI Search Similarity (FAISS) is a commonly used HNSW implementation and you will see it used later in this course
- Constructing the hierarchy and the base network is the hard part of HNSW
- We won't look at it in detail here but, if you're interested:
  - https://www.datastax.com/guides/hierarchical-navigable-small-worlds

Retrieval-Augmented Generation - 6



Source: https://en.wikipedia.org/wiki/Prompt_engineering#/media/File:RAG_schema.svg
By Gknor - Own work, CC BY-SA 4.0, https://commons.wikimedia.org/w/index.php?curid=139459823

Retrieval-Augmented Generation - 7



And responds to the prompt

Source: https://en.wikipedia.org/wiki/Prompt_engineering#/media/File:RAG_schema.svg

RAG Example Notebook (Xilin)

Knowledge Graphs

- A knowledge graph is a data model that uses a graph to organize and represent domain knowledge in the form of entities and relationships
- Knowledge graphs have a formal semantics for
  - Storing knowledge (entities, relationships)
  - Retrieving knowledge (knowledge searching)

Knowledge Graph: Example

Fine Tuning an LLM

Fine Tuning an LLM

- When custom data is added to an LLM using RAG, the model itself is not updated
  - All the parameters stay the same
  - Vectors representing the new knowledge are computed and stored in a database
  - The LLM retrieves these vectors (i.e., the specific data chunks) and combines it with its model (the LLM network parameters) to figure out the output
  - The LLM itself is unchanged
- In fine tuning
  - The model is updated with new parameters
  - You get, in essence, a new LLM
- When fine tuning, you need to go through all the steps in the machine learning process

Fine Tuning: Broad Steps

- Gather data: The quality of the data is the most important input into a model. Data should be representative of the domain you are customizing on; should be sufficient in quantity
- Preprocessing and feature engineering
  - Clean the data
  - Create appropriate features
- Split into training/validation/testing sets: Fine tuning changes model parameters and you need to ensure that the model is learning “correctly”
- Fine tune the model
- Test the fine tuned model

Supervised Fine Tuning: Process

Supervised fine tuning

- The pre-trained LLM is given specific labeled examples
    - A prompt
        - *How can I return my LCD TV?*
    - A response
        - *To return your LED TV, ensure it is in its original packaging and includes all accessories. Returns are accepted within 30 days of purchase. A restocking fee of 15% may apply.*
- The prompt is used to generate a response
        - Example: A generic response on returns
- The generated response is compared with the labeled response
- Weights are adjusted to account for the error
- The process is repeated

Supervised Fine Tuning: Advantages and Disadvantages

- Advantages:
  - Lower processing and memory requirements than full retraining
  - Fall back to pre-trained model is more seamless (compared to RAG)
  - Adapts to the specific domain (like RAG)
- Disadvantages:
  - Model weights are changed and this may compromise the reliability of the LLM for non-domain questions
  - Overfitting: This is a big danger since the model weights are changed. General queries may still give a domain specific response even where they are not suitable
  - Data issues: Data has to be reconstituted in a prompt response format and this may not be practical
    - For example, many different forms of the LCD TV prompt need to prepared
    - And this has to be repeated for all possible prompt/response pairs

Instruction Tuning

- In instruction tuning, the pre-trained LLM is provided with an instruction and given a response
  - Instruction: Translate *I love you* into Italian
  - Response: te amo
- Typical use cases:
  - Language translation
  - Multiple choice tests
- Instruction fine tuning works like supervised fine tuning (the model is updated) but is used
  - when there is insufficient labeled data
  - When the response is well defined

Parameter Efficient Fine Tuning (PEFT)

- Supervised fine tuning and instruction fine tuning update the entire model
- Since an LLM is huge (trillions of parameters), these methods are relatively resource intensive
  - Though less intensive than a complete retraining!
- Parameter efficient fine tuning focuses on updating a small part (subset) of the model
  - Can work with less data (since the training is focused)
  - More efficient (since only a small subset of the model is being changed)
  - Less likely to be overfitted (since the LLM is largely unchanged)
  - However, not as reliable (since only a small subset of the LLM is retrained)
- PEFT is mostly used when
  - Resources are limited
  - Data availability is limited

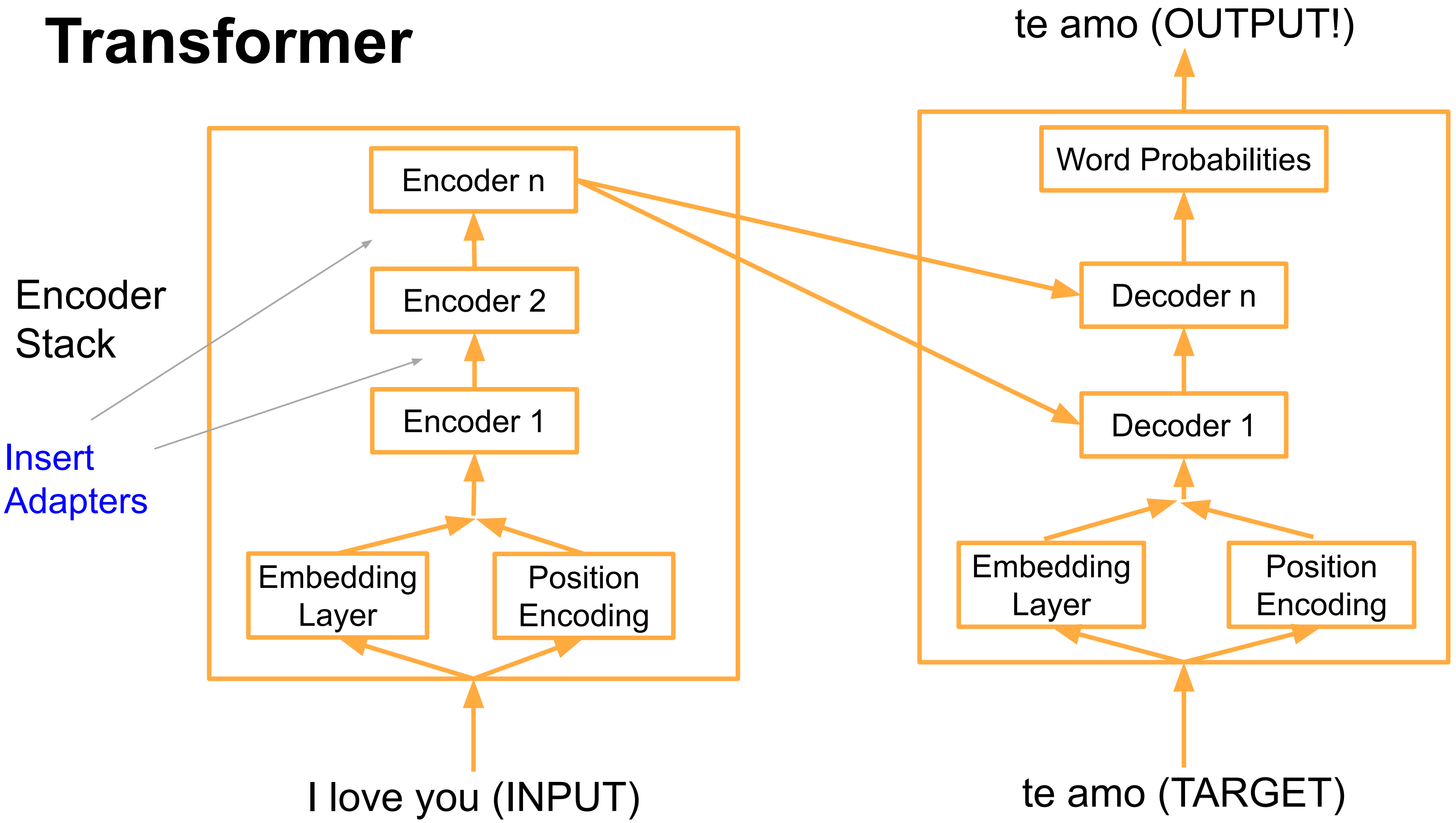
Types of Parameter Efficient Fine Tuning

- Adapters
- Low rank adaption (LORA) and Quantized Low Rank Adaption (QLORA)
- Infused Adapter by Inhibiting and Amplifying Inner Activations (IA3)
- Layer freezing
- Prefix tuning
- Prompt tuning

Adapters

- Adapters are new submodules that are inserted into the transformer architecture
- With each training case (labeled) only the weights in the adapter modules are updated
- The original pre-trained LLM weights are not changed
- Since the adapters are relatively small (few weights) the resources required are relatively low

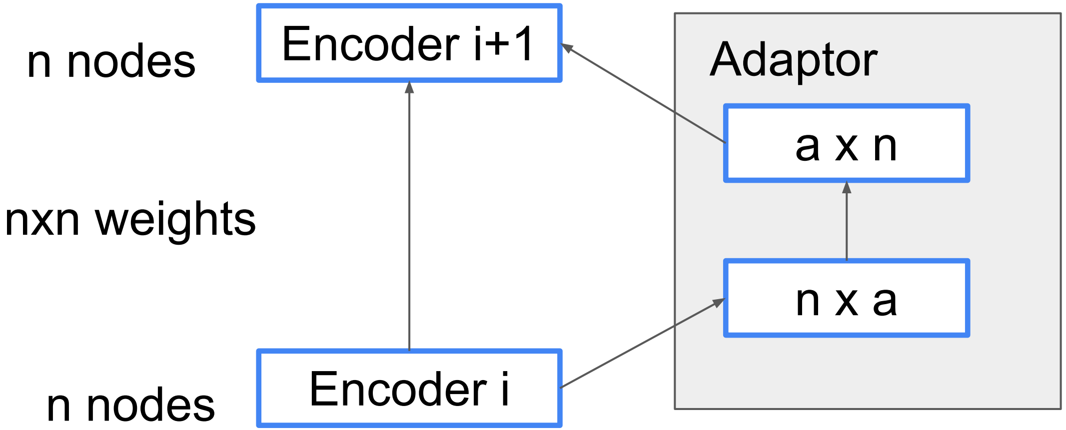
Transformer



LORA: Low Rank Adaptation

- In between any two layers of a transformer, there are n x n weights
- LORA keeps two smaller matrices
  - n x a and a x n, where a << n
- As each case is passed through the transformer
  - The change in weights is computed
  - A lower dimension approximation of this change is used to update the weights in the LORA adaptor
  - The original weights are unchanged
- The main advantage is the reduced memory and processing requirement
  - The LORA adaptor is many (many!) orders of magnitude smaller than the pre-trained LLM

LORA Adaptor



a << n

IA3

- IA3 is structurally similar to LORA
- But, the low rank vectors are directly learned rather than computed from the weight changes of the original model
- This makes IA3 faster and more memory efficient than LORA

Layer Freezing

- Roughly
  - The early layers in a model are more general
    - Language elements
    - General knowledge
  - Later layers are more specialized
    - Domain specific knowledge
    - Derived knowledge
    - How to knowledge (classify, summarize, translate, etc.)
- Layer freezing attempts to freeze early layers and update weights only in later layers when fine tuning a model
- Models can be trained to figure out which layers (or parameters) need to be updated (beyond the scope of our class!)

Prefix Tuning

- A vector is prepended to the model, before the input
- The purpose of the vector is to provide an operational context to the LLM
- For example:
  - A prefix vector may guide the model to produce a summary of the input
  - A prefix vector may guide the model to produce a translation of the input
  - A prefix vector may guide the model to set the context to Olympics
  - A prefix vector may guide the model to set the context to the presidential elections
- Advantages and disadvantages
  - Very memory efficient (a vector) and fast training (only the prefix vector is updated)
  - Limited use since it is setting a context rather than building new information into the LLM

Prompt Tuning

- The same prompt can be written in many different ways
  - How do I return my LCD TV
  - I bought an LCD TV from your store and now realize it is too big for the space and want to return it. What should I do?
- The above two examples ask the same question but in different ways
- In prompt tuning, a preprocessing layer that is trained with sample prompts is inserted between the input and the model
  - The preprocessing layer adds a set of embeddings to the prompt
  - These embeddings direct the prompt (sort of) toward a standard prompt
  - In the two examples above, both prompts will be directed toward the same question
- No new knowledge is added but the model can be guided toward a specific purpose

Fine Tuning Example

How to create an Open AI API Key

Screenshots or b-roll

https://whatsthebigdata.com/how-to-get-openai-api-key/