Building Text-Based Applications with the ChatGPT API & LangChain

O'REILLY®

How to build LLM apps

Lucas Soares

21-08-2023

Intro

Hi!

- Me!
- Quick Survey to get to know everyone!

Quick Survey!

Where are you from?

Answer in the Chat with: City, Country (initials)

Methodology Notes

1. Use emojis to demonstrate understanding (or lack of it)

Methodology Notes

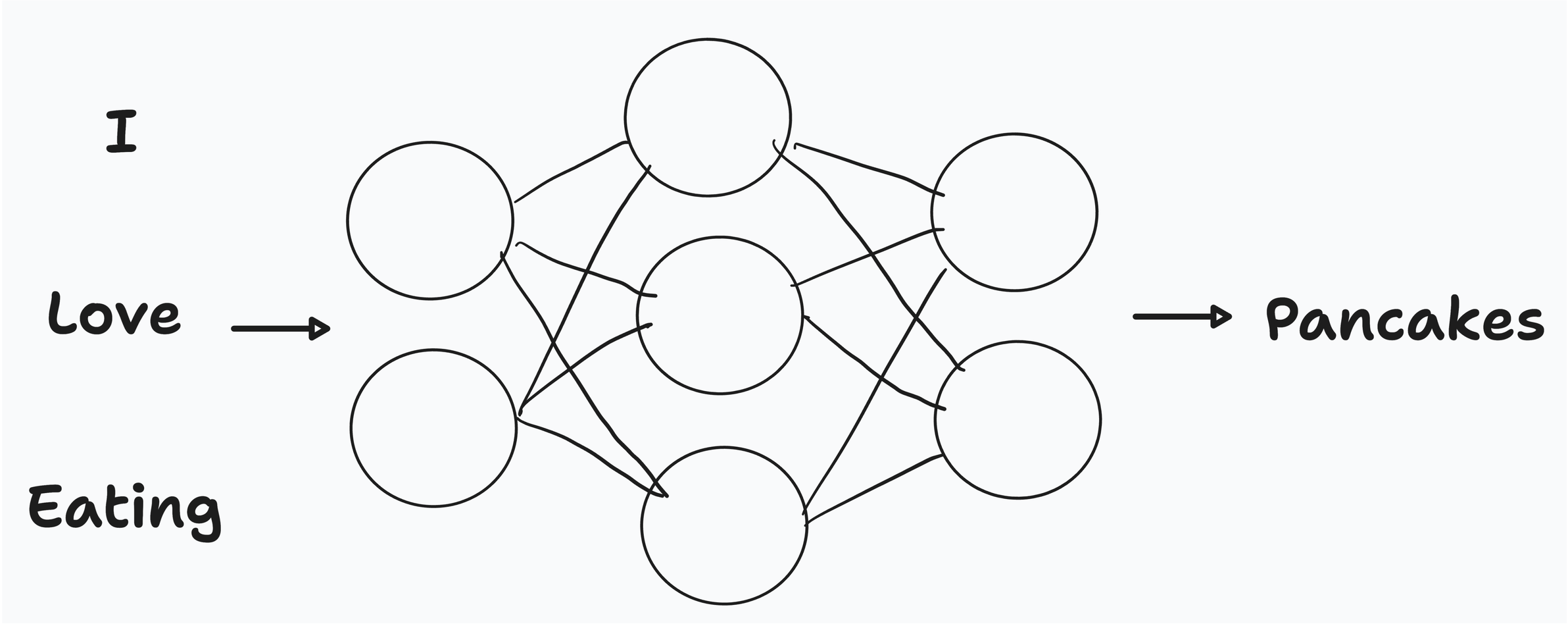
1. Use emojis to demonstrate understanding (or lack of it)
2. Ask questions! (Q&A - technical, chat - quick ones)

Methodology Notes

1. Use emojis to demonstrate understanding (or lack of it)
2. Ask questions! (Q&A - technical, chat - quick ones)
3. Engage and participate! (This live-training is for you!)

Large Language Models

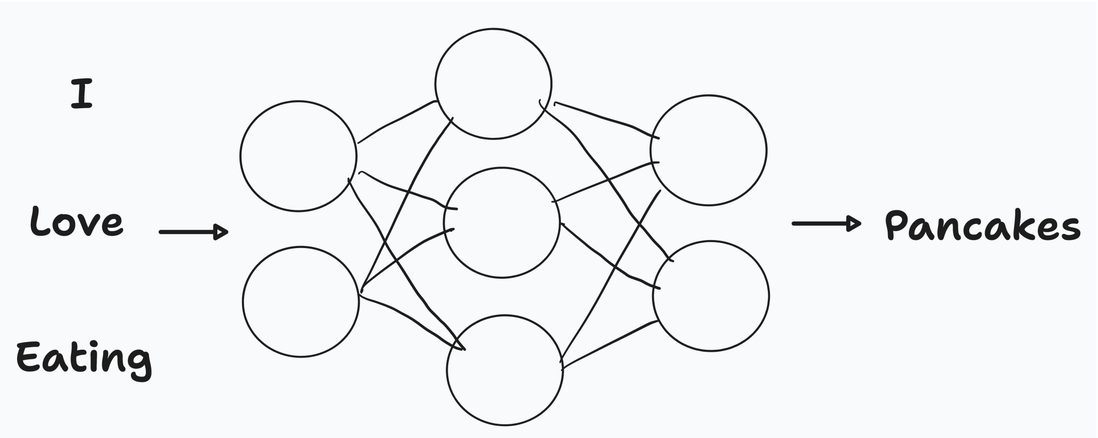
A definition



Large Language Models

As Probability Distributions

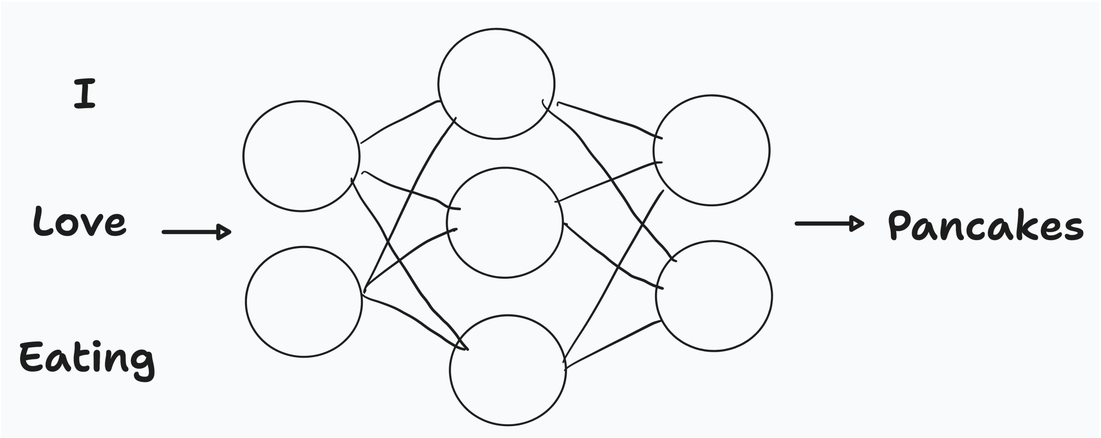
At their core, LLMs can be seen as distributions over words.



Large Language Models

As Probability Distributions

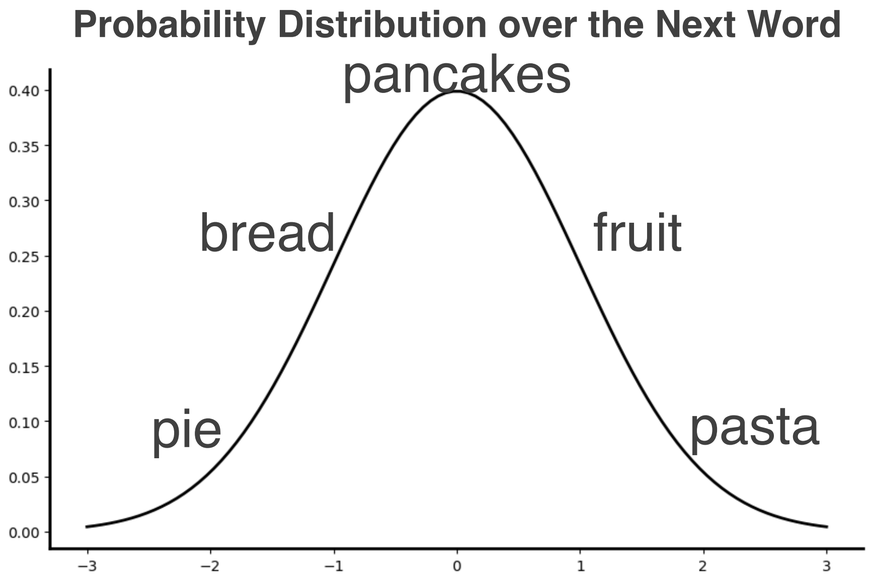
- At their core, LLMs can be seen as distributions over words.
- Use statistical models to capture patterns in text data.



Large Language Models

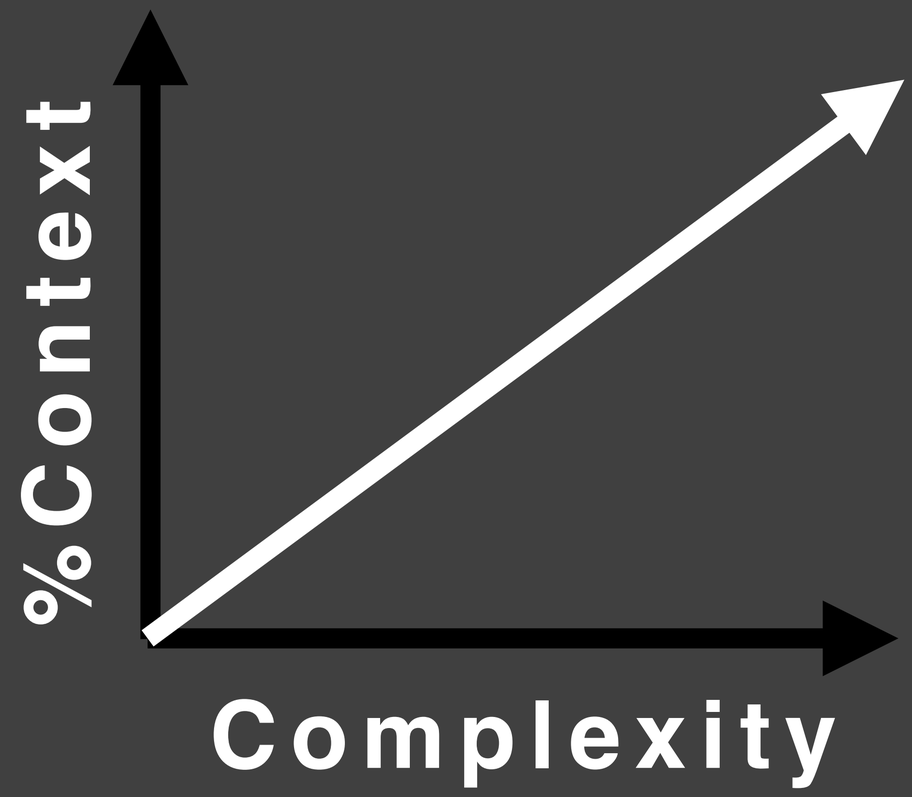
As Probability Distributions

- At their core, LLMs can be seen as distributions over words.
- Use statistical models to capture patterns in text data.
- They calculate the likelihood of each word occurring given the context.



"I love eating…." → ?

They capture context, understand dependencies, and predict text based on patterns learned during training.


%Context
Complexity

Ok, but how?

LLMs as pattern matchers

LLMs are akin to probabilistic programs.

Ok, but how?

LLMs as pattern matchers

- LLMs are akin to probabilistic programs.
- They predict outcomes based on probabilities learned from training data

Ok, but how?

LLMs as pattern matchers

- LLMs are akin to probabilistic programs.
- They predict outcomes based on probabilities learned from training data

```python
import numpy as np

def llm_model(context, next_word):
    possible_next_words = ["text", "pie", "pizza", "motor", "cake"]
    next_word_probs = [0.2, 0.3, 0.4, 0.001, 0.6]
    return next_word_probs[possible_next_words.index(next_word)]

context  = ["This", "is", "a", "piece", "of"]
possible_next_words = ["text", "pie", "pizza", "motor", "cake"]
probs = []
for w in possible_next_words:
    probs.append(llm_model(context, w))

next_word = possible_next_words[np.argmax(probs)]
next_wor
```


Context/Past
LM
Future

How LLMs work

Transformers Architecture

- Traditional sequential models struggle with context (Vaswani et al 2017)

How LLMs work

Transformers Architecture

- Traditional sequential models struggle with context (Vaswani et al 2017)
- Transformers use attention mechanisms to capture global dependencies, enabling contextual understanding. (Vaswani et al 2017)

How LLMs work

Transformers Architecture

- Traditional sequential models struggle with context (Vaswani et al 2017)
- Transformers use attention mechanisms to capture global dependencies, enabling contextual understanding. (Vaswani et al 2017)
- The attention mechanism allows Transformers to focus on different parts of input simultaneously. (Vaswani et al 2017)

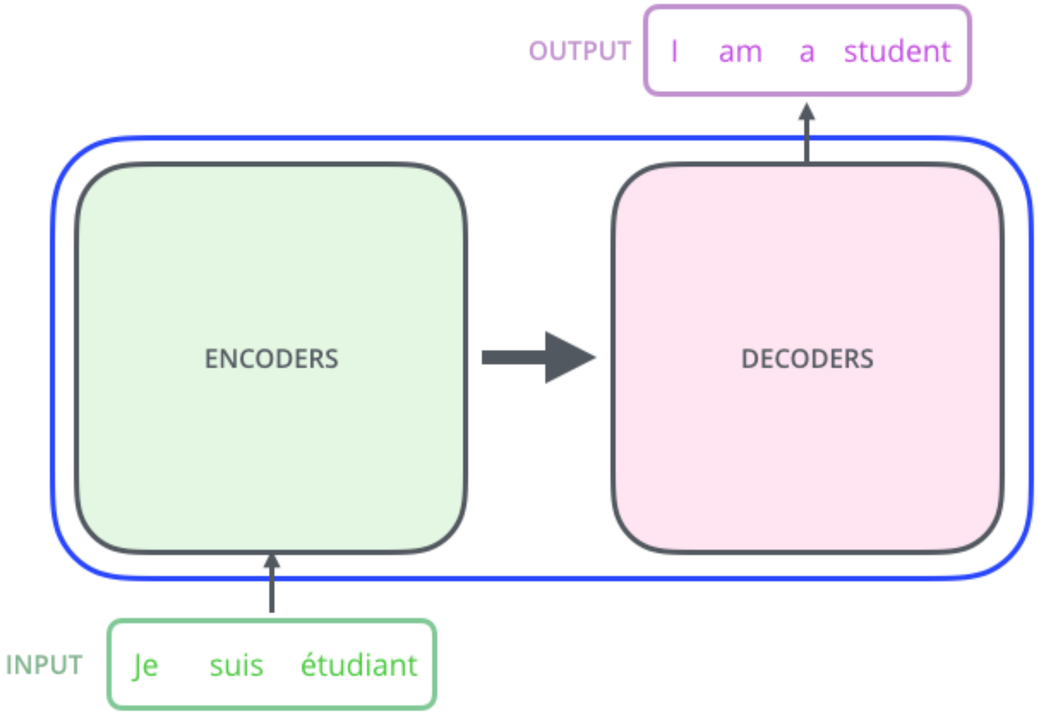
How LLMs work

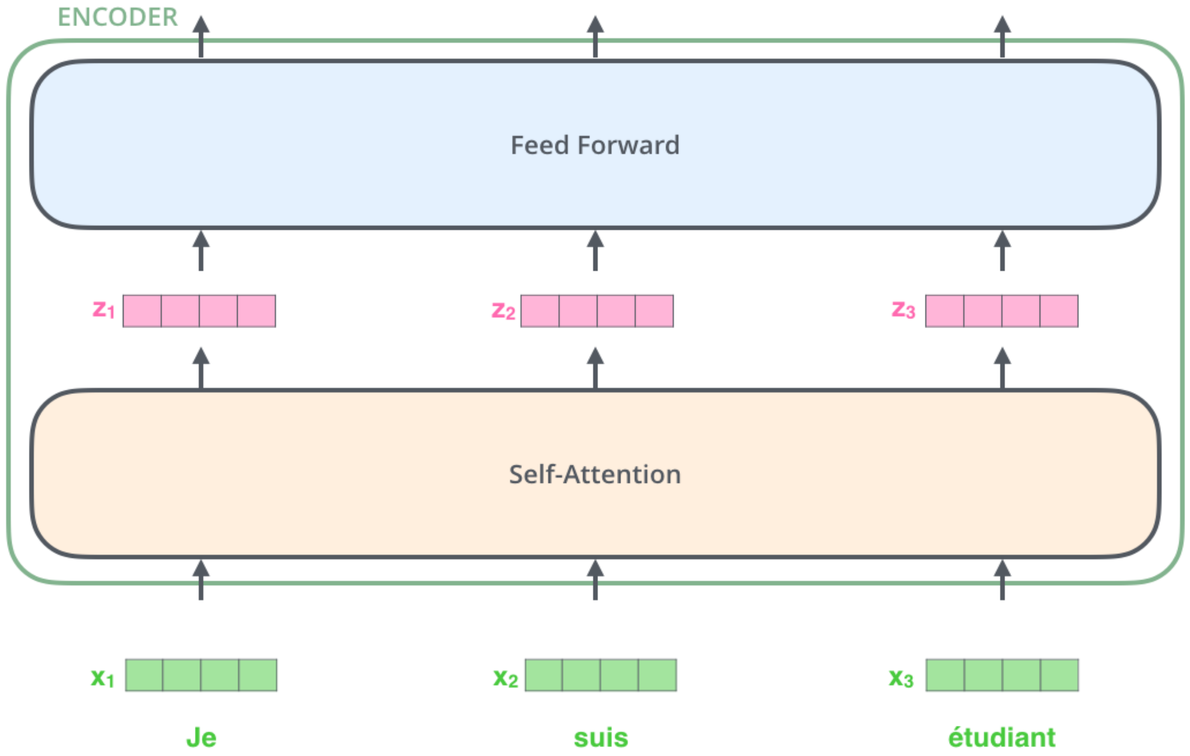
Transformers Architecture

- Traditional sequential models struggle with context (Vaswani et al 2017)
- Transformers use attention mechanisms to capture global dependencies, enabling contextual understanding. (Vaswani et al 2017)
- The attention mechanism allows Transformers to focus on different parts of input simultaneously. (Vaswani et al 2017)
- Transformers can understand and predict based on long-range dependencies. (Vaswani et al 2017)

How LLMs work

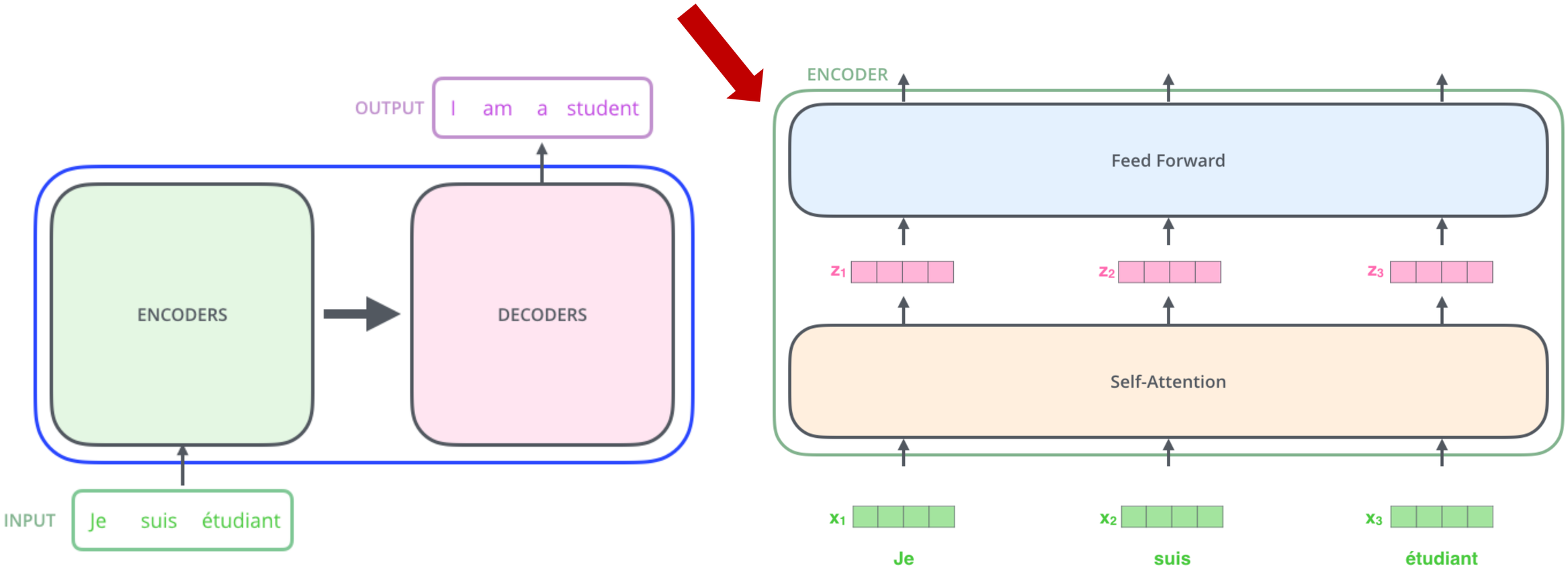
Transformers Architecture





How LLMs work

Transformers Architecture



Inputs are processed in parallel!

Benefits of Large Language Models

Multi-task, fine tuning, scalability

- **Multi-task Capability:** LLMs find applications in content generation, question answering, translation, tutoring, and personal assistants.

Benefits of Large Language Models

Multi-task, fine tuning, scalability

- **Multi-task Capability:** LLMs find applications in content generation, question answering, translation, tutoring, and personal assistants.
- **Fine-tunning:** LLMs can usually be fine tuned with a relatively small amount of data, making them adaptable to a wide range of tasks.

Benefits of Large Language Models

Multi-task, fine tuning, scalability

- **Multi-task Capability:** LLMs find applications in content generation, question answering, translation, tutoring, and personal assistants.
- **Fine-tunning:** LLMs can usually be fine tuned with a relatively small amount of data, making them adaptable to a wide range of tasks.
- **Scalability**: LLMs demonstrate excellent scalability to very large capacity networks and huge datasets.

Applications of Large Language Models

- Content generation
- Q&A
- Translation
- Tutoring
- Personal Assistants

Applications of LLMs

Content generation Example - DEMO

Applications of LLMs

Translation Example - DEMO

Limitations and Ethical Considerations

LLMs are far from perfect

- **Knowledge Limit:** LLMs have a cutoff point for their knowledge.

Limitations and Ethical Considerations

LLMs are far from perfect

- **Knowledge Limit:** LLMs have a cutoff point for their knowledge.
- **Understanding Limit**: LLMs do not understand text in the same way humans do. They don't have beliefs or desires; they simply predict what comes next based on their training.

Limitations and Ethical Considerations

LLMs are far from perfect

- **Knowledge Limit:** LLMs have a cutoff point for their knowledge.
- **Understanding Limit**: LLMs do not understand text in the same way humans do. They don't have beliefs or desires; they simply predict what comes next based on their training.
- **Misuse:** LLMs can hallucinate and produce false or harmful content

Limitations and Ethical Considerations

LLMs are far from perfect

- **Knowledge Limit:** LLMs have a cutoff point for their knowledge.
- **Understanding Limit**: LLMs do not understand text in the same way humans do. They don't have beliefs or desires; they simply predict what comes next based on their training.
- **Misuse:** LLMs can hallucinate and produce false or harmful content
- **Reproducibility**: Unpredictability of LLM behavior. Watkins 2023

Limitations and Ethical Considerations

LLMs are far from perfect

- **Knowledge Limit:** LLMs have a cutoff point for their knowledge.
- **Understanding Limit**: LLMs do not understand text in the same way humans do. They don't have beliefs or desires; they simply predict what comes next based on their training.
- **Misuse:** LLMs can hallucinate and produce false or harmful content
- **Reproducibility**: Unpredictability of LLM behavior. Watkins 2023
- **Data Privacy and Bias:** Ethical considerations should extend to the acquisition of data for training additional models. Models may have biases; their use should be transparent and biases mitigated. Watkins 2023

Q & A

Notebook Demo

Introduction to prompt engineering and the ChatGPT API

1. Prompt basics
2. Introduction to the ChatGPT API
3. Prompt engineering guide
4. Demos & Exercises

1

Prompt Basics

Introduction to prompts

A prompt is a piece of text that conveys to the LLM the user’s intention.

1

Prompt Basics

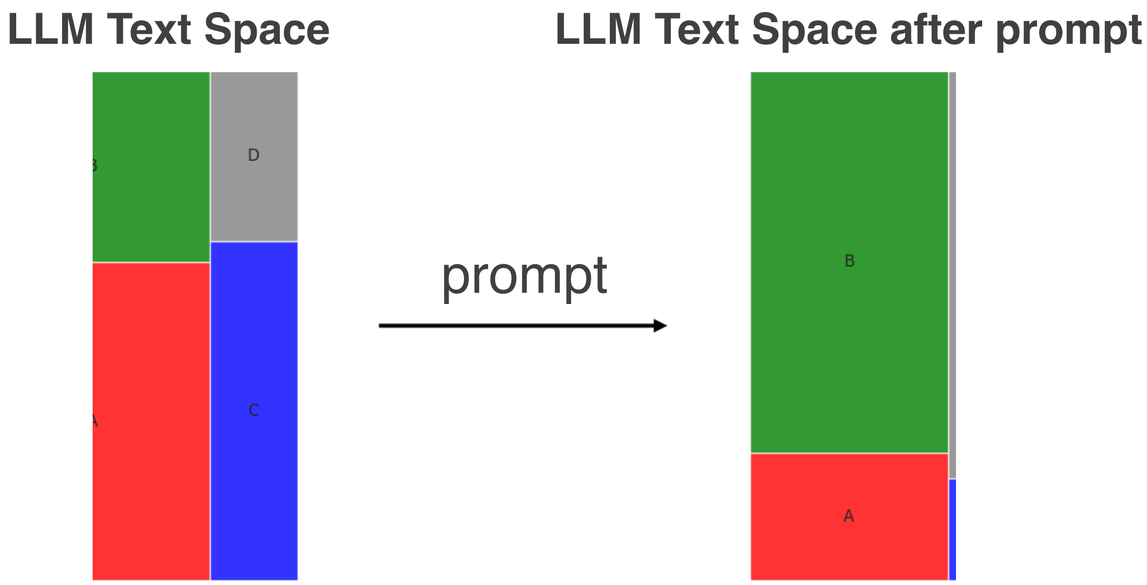
Introduction to prompts

- A prompt is a piece of text that conveys to the LLM the user’s intention.
- Question → Instruction → Behavior

Prompt Basics

Introduction to prompts

- A prompt is a piece of text that conveys to the LLM the user’s intention.
- Question → Instruction → Behavior
- It **constrains the space of possibilities** in the LLM's text space



1

Prompt Basics

Components of a prompt: instruction, context, input data, output indicator

Prompt Basics

Components of a prompt: instruction, context, input data, output indicator

You are a text annotation engine.

Classify this sentence into positive or negative:
Text: *I like eating pancakes*
Output:

Prompt Basics

Components of a prompt: instruction, context, input data, output indicator

Instruction: where you describe what you want

You are a text annotation engine.
Classify this sentence into positive or negative:
Text: *I like eating pancakes*
Output:

Prompt Basics

Components of a prompt: instruction, context, input data, output indicator

- **Instruction:** where you describe what you want
- **Context:** additional information to help with performance.

You are a text annotation engine.

Classify this sentence into positive or negative:
Text: ***I like eating pancakes***
Output:

Prompt Basics

Components of a prompt: instruction, context, input data, output indicator

- **Instruction:** where you describe what you want
- **Context:** additional information to help with performance.
- **Input data:** example of data you wish the model to process

You are a text annotation engine.

Classify this sentence into positive or negative:
Text: ***I like eating pancakes***
Output:

Prompt Basics

Components of a prompt: instruction, context, input data, output indicator

- **Instruction:** where you describe what you want
- **Context:** additional information to help with performance.
- **Input data:** example of data you wish the model to process
- **Output indicator:** Priming model to output what you want (e.g structure, length, etc..)

You are a text annotation engine.

Classify this sentence into positive or negative:
Text: ***I like eating pancakes***
Output:

Prompt Basics

Components of a prompt: instruction, context, input data, output indicator

- **Instruction:** where you describe what you want
- **Context:** additional information to help with performance.
- **Input data:** example of data you wish the model to process
- **Output indicator:** Priming model to output what you want (e.g structure, length, etc..)

Notebook Demo

Introduction to the ChatGPT API

Where does ChatGPT fit in this chaotic LLM universe?

- The ChatGPT API allows us to use OpenAI's models to generate dynamic, contextually-aware responses.
- Required parameters: model, messages.

```python
from IPython.display import Markdown
from openai import OpenAI
client = OpenAI()

def get_response(prompt_question):
    response = client.chat.completions.create(
        model="gpt-3.5-turbo-16k",
        messages=[{"role": "system", "content": "You are a helpful research\
            and programming assistant"},
                  {"role": "user", "content": prompt_question}]
    )

    return response.choices[0].message.content

prompt = "Hi! Tell me a joke about large language models."
Markdown(get_response(prompt))
```

✓ 0.8s

Sure, here's one for you:

Why did the large language model go on a diet?

Because it was worried about gaining too many bytes!

Introduction to the ChatGPT API

Where does ChatGPT fit in this chaotic LLM universe?

```
# !pip install openai
import openai
import os
openai.api_key = os.environ["OPENAI_API_KEY"]
from IPython.display import Markdown
from openai import OpenAI
client = OpenAI()

def get_response(prompt_question):
    response = client.chat.completions.create(
        model="gpt-3.5-turbo-16k",
        messages=[{"role": "system", "content": "You are a helpful research\
            and programming assistant"},
                {"role": "user", "content": prompt_question}]
    )

    return response.choices[0].message.content

prompt = "Hi! Tell me a joke about large language models."
Markdown(get_response(prompt))
```

Notebook Demo - Intro ChatGPT API

Prompt Engineering Guide

What is prompt engineering?

Prompt Engineering Guide

What is prompt engineering?

Prompt engineering: discipline for engineering prompts

Prompt Engineering Guide

What is prompt engineering?

- **Prompt engineering:** discipline for engineering prompts
- The basic goal of prompt engineering is **designing good prompts.**

Prompt Engineering Guide

What is prompt engineering?

- **Prompt engineering:** Discipline for engineering prompts
- The basic goal of prompt engineering is **designing good prompts.**

It’s about **having a process for developing good prompts** that yield high performance across tasks.

Prompt Engineering Guide

What is prompt engineering?

- **Prompt engineering:** Discipline for engineering prompts

- The basic goal of prompt engineering is **designing good prompts.**

It’s about **having a process for developing good prompts** that yield high performance across tasks.

Prompt Engineering Guide

OpenAI's Guide for Building Good Prompts

Prompt Engineering Guide

OpenAI's Guide for Building Good Prompts

Strategy 1: Write clear instructions

[OpenAI docs]

Prompt Engineering Guide

OpenAI's Guide for Building Good Prompts

Strategy 1: Write clear instructions

Bad: Who's president?
Better: Who was the president of Mexico in 2021?

OpenAI docs

Prompt Engineering Guide

OpenAI's Guide for Building Good Prompts

Strategy 1: Write clear instructions

Strategy 2: Provide reference text

OpenAI docs

Prompt Engineering Guide

OpenAI's Guide for Building Good Prompts

- **Strategy 1:** Write clear instructions
- **Strategy 2:** Provide reference text

SYSTEM
Use the provided articles delimited by triple quotes to answer questions. If the answer cannot be found in the articles, write "I could not find an answer."
USER
<insert articles, each delimited by triple quotes>

Question: <insert question here>

Prompt Engineering Guide

OpenAI's Guide for Building Good Prompts

Strategy 1: Write clear instructions

Strategy 2: Provide reference text

Strategy 3: Break tasks into subtasks

OpenAI docs

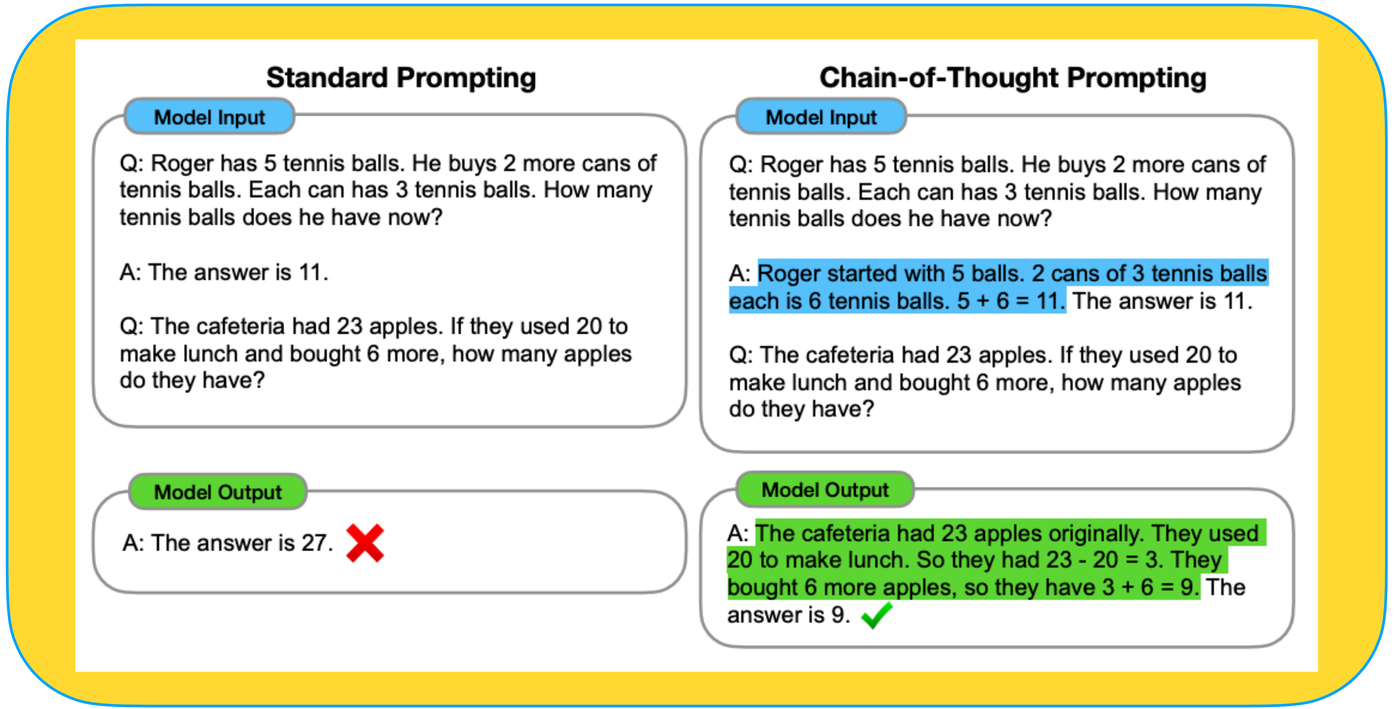
Prompt Engineering Guide

OpenAI's Guide for Building Good Prompts

Strategy 1: Write clear instructions

Strategy 2: Provide reference text

Strategy 3: Break tasks into subtasks



Prompt Engineering Guide

OpenAI's Guide for Building Good Prompts

- **Strategy 1:** Write clear instructions
- **Strategy 2:** Provide reference text
- **Strategy 3:** Break tasks into subtasks
- **Strategy 4:** Give the model time to think

[OpenAI docs]

Prompt Engineering Guide

OpenAI’s Guide for Building Good Prompts

- **Strategy 1:** Write clear instructions
- **Strategy 2:** Provide reference text
- **Strategy 3:** Break tasks into subtasks
- **Strategy 4:** Give the model time to think

SYSTEM
Follow these steps to answer the user queries.

Step 1 - First work out your own solution to the problem. Don't rely on the student's solution since it may be incorrect. Enclose all your work for this step within triple quotes (""").

Step 2 - Compare your solution to the student's solution and evaluate if the student's solution is correct or not. Enclose all your work for this step within triple quotes (""").

Step 3 - If the student made a mistake, determine what hint you could give the student without giving away the answer. Enclose all your work for this step within triple quotes (""").

Step 4 - If the student made a mistake, provide the hint from the previous step to the student (outside of triple quotes). Instead of writing "Step 4 - ..." write "Hint:".
USER
Problem Statement: <insert problem statement>

Student Solution: <insert student solution>

Prompt Engineering Guide

OpenAI's Guide for Building Good Prompts

- **Strategy 1:** Write clear instructions
- **Strategy 2:** Provide reference text
- **Strategy 3:** Break tasks into subtasks
- **Strategy 4:** Give the model time to think
- **Strategy 5:** Use external tools

[OpenAI docs]

Prompt Engineering Guide

OpenAI's Guide for Building Good Prompts

- **Strategy 1:** Write clear instructions
- **Strategy 2:** Provide reference text
- **Strategy 3:** Break tasks into subtasks
- **Strategy 4:** Give the model time to think
- **Strategy 5:** Use external tools

SYSTEM
You can write and execute Python code by enclosing it in triple backticks, e.g. ```code goes here```. Use this to perform calculations.
USER
Find all real-valued roots of the following polynomial: 3*x**5 - 5*x**4 - 3*x**3 - 7*x - 10.

Prompt Engineering Guide

OpenAI's Guide for Building Good Prompts

Strategy 1: Write clear instructions

Strategy 2: Provide reference text

Strategy 3: Break tasks into subtasks

Strategy 4: Give the model time to think

Strategy 5: Use external tools

Strategy 6: Test changes systematically

OpenAI docs

Prompt Engineering Guide

OpenAI's Guide for Building Good Prompts

- **Strategy 1:** Write clear instructions
- **Strategy 2:** Provide reference text
- **Strategy 3:** Break tasks into subtasks
- **Strategy 4:** Give the model time to think
- **Strategy 5:** Use external tools
- **Strategy 6:** Test changes systematically

Evaluation procedures are useful for optimizing system designs. **Good evals are:**

- **Representative** of real-world usage (or at least diverse)
- **Contain many test cases** for greater statistical power (see table below for guidelines)
- **Easy to automate** or repeat

Prompt Engineering Guide

OpenAI's Guide for Building Good Prompts

- **Strategy 1:** Write clear instructions
- **Strategy 2:** Provide reference text
- **Strategy 3:** Break tasks into subtasks
- **Strategy 4:** Give the model time to think
- **Strategy 5:** Use external tools
- **Strategy 6:** Test changes systematically

Evaluation procedures are useful for optimizing system designs. **Good evals are:**

- **Representative** of real-world usage (or at least diverse)
- **Contain many test cases** for greater statistical power (see table below for guidelines)
- **Easy to automate** or repeat

Notebook Demo - Applying Prompt Engineering Strategies

Q&A / Break

Fine Tuning ChatGPT Applications

A primer on how to fine tune ChatGPT on your data

Fine Tuning:

Fine Tuning ChatGPT Applications

A primer on how to fine tune ChatGPT on your data

Fine Tuning: process of “specialising an LLM on your specific dataset”

Fine Tuning ChatGPT Applications

A primer on how to fine tune ChatGPT on your data

Fine Tuning: process of “specialising an LLM on your specific dataset”

Why?

Fine Tuning ChatGPT Applications

A primer on how to fine tune ChatGPT on your data

Fine Tuning: process of “specialising an LLM on your specific dataset”

Why? Better results on specific tasks, lower latency… **See OpenAI docs**

Fine Tuning ChatGPT Applications

A primer on how to fine tune ChatGPT on your data

Fine Tuning: process of “specialising an LLM on your specific dataset”

Why? Better results on specific tasks, lower latency… **See OpenAI docs**

How?

Fine Tuning ChatGPT Applications

A primer on how to fine tune ChatGPT on your data

Fine Tuning: process of “specialising an LLM on your specific dataset”

Why? Better results on specific tasks, lower latency… **See OpenAI docs**

How? By preparing a custom `.jsonl` dataset on your specific use case

Fine Tuning ChatGPT Applications

A primer on how to fine tune ChatGPT on your data

Fine Tuning: process of “specialising an LLM on your specific dataset”

Why? Better results on specific tasks, lower latency… **See OpenAI docs**

How? By preparing a custom `.jsonl` dataset on your specific use case

The goal of fine tuning is to adapt the model to your specific use case by providing it with a few examples of successful interactions

3

Fine Tuning ChatGPT Applications

Common Use Cases

Fine Tuning ChatGPT Applications

Common Use Cases

- Setting style, tone, format

[OpenAI docs]

Fine Tuning ChatGPT Applications

Common Use Cases

- Setting style, tone, format
- Correcting failures

[OpenAI docs]

Fine Tuning ChatGPT Applications

Common Use Cases

- Setting style, tone, format
- Correcting failures
- Improving reliability

OpenAI docs

Fine Tuning ChatGPT Applications

Common Use Cases

- Setting style, tone, format
- Correcting failures
- Improving reliability
- Handling edge cases

OpenAI docs

Fine Tuning ChatGPT Applications

Common Use Cases

- Setting style, tone, format
- Correcting failures
- Improving reliability
- Handling edge cases
- Extracting structured output

OpenAI docs

Fine Tuning ChatGPT Applications

Common Use Cases

- Setting style, tone, format
- Correcting failures
- Improving reliability
- Handling edge cases
- Extracting structured output
- Function Calling

OpenAI docs

Fine Tuning ChatGPT Applications

Basic Steps

OpenAI docs

Fine Tuning ChatGPT Applications

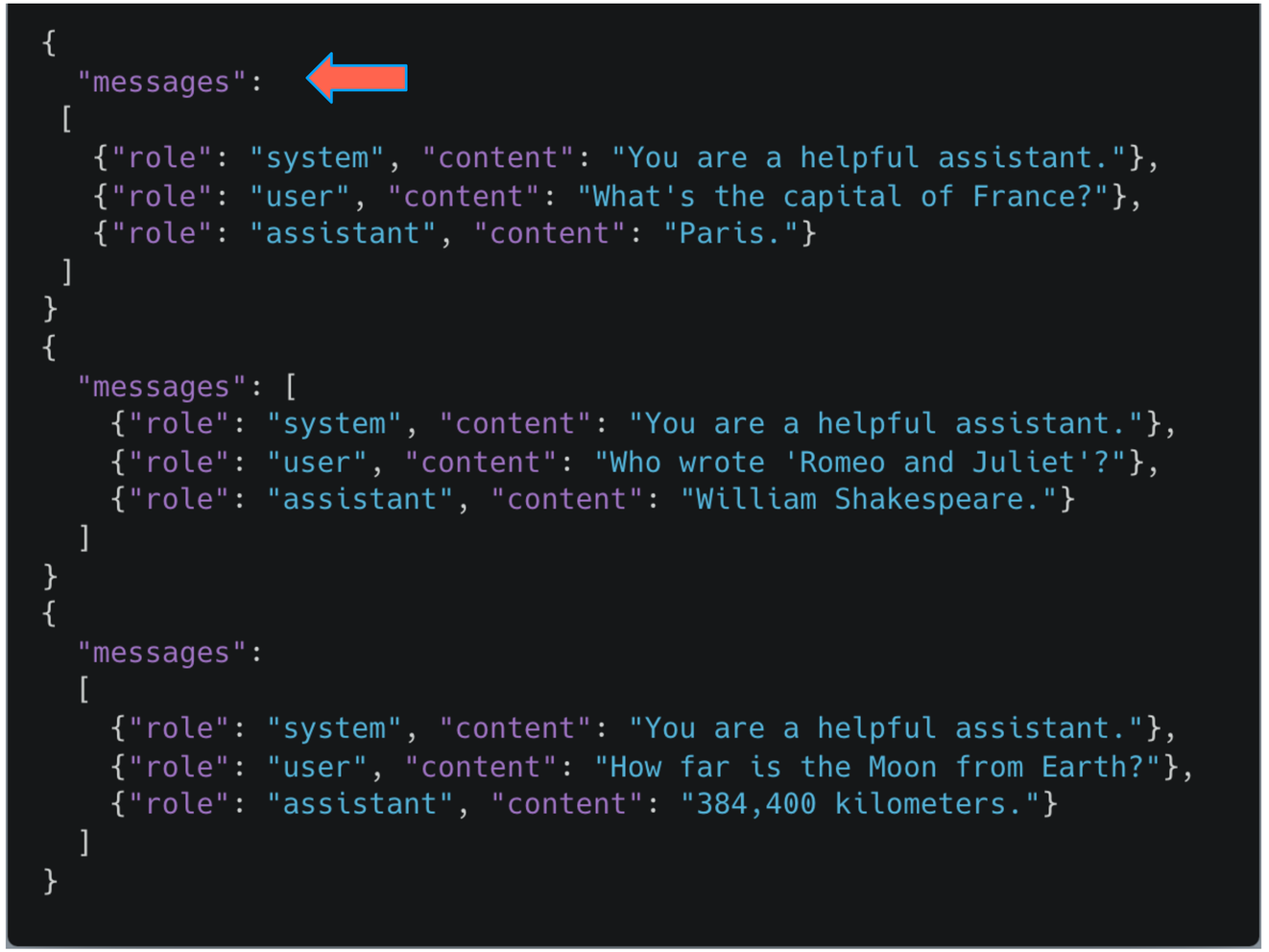
Basic Steps

- Prepare and upload training data

[OpenAI docs]

Fine Tuning ChatGPT Applications

Prepare and upload training data

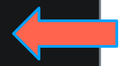


```
{
  "messages":
 [
   {"role": "system", "content": "You are a helpful assistant."},
   {"role": "user", "content": "What's the capital of France?"},
   {"role": "assistant", "content": "Paris."}
 ]
}
{
  "messages": [
    {"role": "system", "content": "You are a helpful assistant."},
    {"role": "user", "content": "Who wrote 'Romeo and Juliet'?"},
    {"role": "assistant", "content": "William Shakespeare."}
  ]
}
{
  "messages":
  [
    {"role": "system", "content": "You are a helpful assistant."},
    {"role": "user", "content": "How far is the Moon from Earth?"},
    {"role": "assistant", "content": "384,400 kilometers."}
  ]
}
```

Fine Tuning ChatGPT Applications

Prepare and upload training data

```
{
  "messages":
 [
   {"role": "system", "content": "You are a helpful assistant."},
   {"role": "user", "content": "What's the capital of France?"},
   {"role": "assistant", "content": "Paris."}
 ]
}
{
  "messages": [
    {"role": "system", "content": "You are a helpful assistant."},
    {"role": "user", "content": "Who wrote 'Romeo and Juliet'?"},
    {"role": "assistant", "content": "William Shakespeare."}
  ]
}
{
  "messages":
  [
    {"role": "system", "content": "You are a helpful assistant."},
    {"role": "user", "content": "How far is the Moon from Earth?"},
    {"role": "assistant", "content": "384,400 kilometers."}
  ]
}
```



Fine Tuning ChatGPT Applications

Prepare and upload training data

```
{
  "messages":
 [
   {"role": "system", "content": "You are a helpful assistant."},
   {"role": "user", "content": "What's the capital of France?"},
   {"role": "assistant", "content": "Paris."}
 ]
}
{
  "messages": [
    {"role": "system", "content": "You are a helpful assistant."},
    {"role": "user", "content": "Who wrote 'Romeo and Juliet'?"},
    {"role": "assistant", "content": "William Shakespeare."}
  ]
}
{
  "messages":
  [
    {"role": "system", "content": "You are a helpful assistant."},
    {"role": "user", "content": "How far is the Moon from Earth?"},
    {"role": "assistant", "content": "384,400 kilometers."}
  ]
}
```

Fine Tuning ChatGPT Applications

Prepare and upload training data

```
{
  "messages":
 [
   {"role": "system", "content": "You are a helpful assistant."},
   {"role": "user", "content": "What's the capital of France?"},
   {"role": "assistant", "content": "Paris."}
 ]
}
{
  "messages": [
    {"role": "system", "content": "You are a helpful assistant."},
    {"role": "user", "content": "Who wrote 'Romeo and Juliet'?"},
    {"role": "assistant", "content": "William Shakespeare."}
  ]
}
{
  "messages":
  [
    {"role": "system", "content": "You are a helpful assistant."},
    {"role": "user", "content": "How far is the Moon from Earth?"},
    {"role": "assistant", "content": "384,400 kilometers."}
  ]
}
```

Fine Tuning ChatGPT Applications

Prepare and upload training data

```
{
  "messages":
 [
   {"role": "system", "content": "You are a helpful assistant."},
   {"role": "user", "content": "What's the capital of France?"},
   {"role": "assistant", "content": "Paris."}
 ]
}
{
  "messages": [
    {"role": "system", "content": "You are a helpful assistant."},
    {"role": "user", "content": "Who wrote 'Romeo and Juliet'?"},
    {"role": "assistant", "content": "William Shakespeare."}
  ]
}
{
  "messages":
  [
    {"role": "system", "content": "You are a helpful assistant."},
    {"role": "user", "content": "How far is the Moon from Earth?"},
    {"role": "assistant", "content": "384,400 kilometers."}
  ]
}
```

Fine Tuning ChatGPT Applications

Prepare and upload training data

```
{
  "messages":
 [
   {"role": "system", "content": "You are a helpful assistant."},
   {"role": "user", "content": "What's the capital of France?"},
   {"role": "assistant", "content": "Paris."}
 ]
}
{
  "messages": [
    {"role": "system", "content": "You are a helpful assistant."},
    {"role": "user", "content": "Who wrote 'Romeo and Juliet'?"},
    {"role": "assistant", "content": "William Shakespeare."}
  ]
}
{
  "messages":
  [
    {"role": "system", "content": "You are a helpful assistant."},
    {"role": "user", "content": "How far is the Moon from Earth?"},
    {"role": "assistant", "content": "384,400 kilometers."}
  ]
}
```

Fine Tuning ChatGPT Applications

Basic Steps

- Prepare and upload training data
- Train a new fine-tuned model

OpenAI docs

Fine Tuning ChatGPT Applications

Fine Tuning Jobs

```python
from openai import OpenAI
# OpenAI API key should be set as
# environment variable - OPENAI_API_KEY
client = OpenAI()
client.files.create(
  file=open("dataset.jsonl", "rb"),
  purpose="fine-tune"
)
client.fine_tuning.jobs.create(
  training_file="file-id",
  model="gpt-3.5-turbo-1106"
)
```

Notebook Presentation Fine Tuning with ChatGPT API

Fine Tuning ChatGPT Applications

Basic Steps

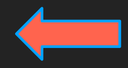
- Prepare and upload training data
- Train a new fine-tuned model
- Use your fine-tuned model

OpenAI docs

Fine Tuning ChatGPT Applications

Fine Tuning Jobs

```python
def get_response(prompt, fine_tuned_model_id="ft:gpt-3.5-turbo-1106:personal::8Oq91T4e"):
    client = OpenAI()
    response = client.chat.completions.create(model=fine_tuned_model_id,
                                              messages=
                                              [
                                                  {"role": "system", "content": "You are a helpful assistant."},
                                                  {"role": "user", "content": prompt}
                                              ],
                                              temperature=0.0,
                                              n = 1
                                              )
    return response.choices[0].message.content
```



Disclaimer: Example code is provided in the repository notebooks

Summary so far

Break

Langchain for LLM App Development

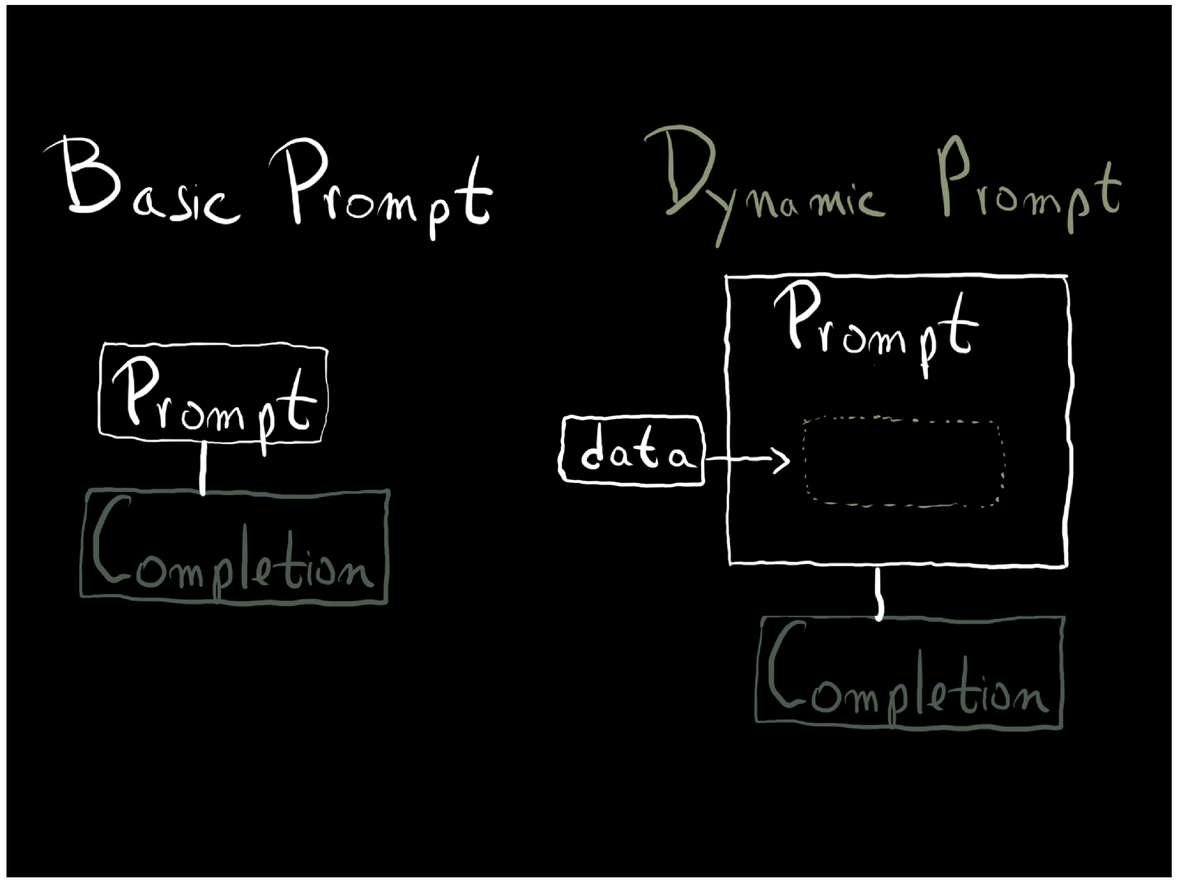
From static prompts to dynamic prompts

- Prompt management workflows that require dynamic prompts

Langchain for LLM App Development

From static prompts to dynamic prompts

- Prompt management workflows that require **dynamic prompts**
- This dynamics requirement leads to the **need for LLM-specific abstractions**



Langchain for LLM App Development

Langchain framework

LangChain

Basics of Langchain

Langchain components

- **Models:** abstractions over the LLM APIs like the ChatGPT API

Basics of Langchain

Langchain components

- **Models:** abstractions over the LLM APIs like the ChatGPT API
- **Prompt Templates:** abstraction over standard prompts to LLMs

Basics of Langchain

Langchain components

- **Models:** abstractions over the LLM APIs like the ChatGPT API
- **Prompt Templates:** abstraction over standard prompts to LLMs
- **Output Parsers:** Translates raw output from LLM to a workable format

https://python.langchain.com/docs/get_started/quickstart

Basics of Langchain

Langchain components

- **Models:** abstractions over the LLM APIs like the ChatGPT API
- **Prompt Templates:** abstraction over standard prompts to LLMs
- **Output Parsers:** Translates raw output from LLM to a workable format

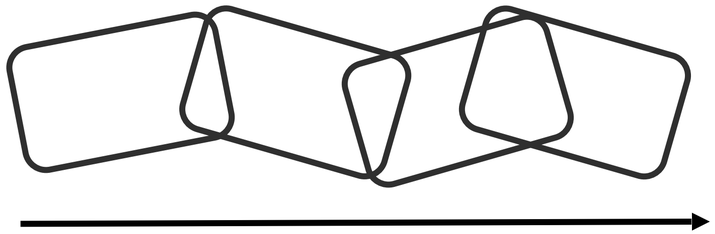
Notebook demo: Introduction to LangChain

https://python.langchain.com/docs/get_started/quickstart

Basics of Langchain

Models, Prompt Templates and Output Parsers

- **Chains:** prompt template + Model + output parsing



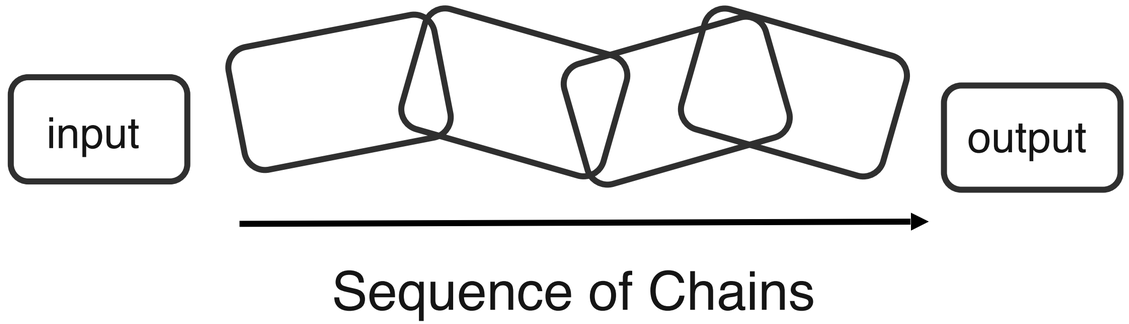
Chains of Prompts

Basics of Langchain

LLMChain & Sequential Chains

- **Chains:** prompt template + Model + output parsing

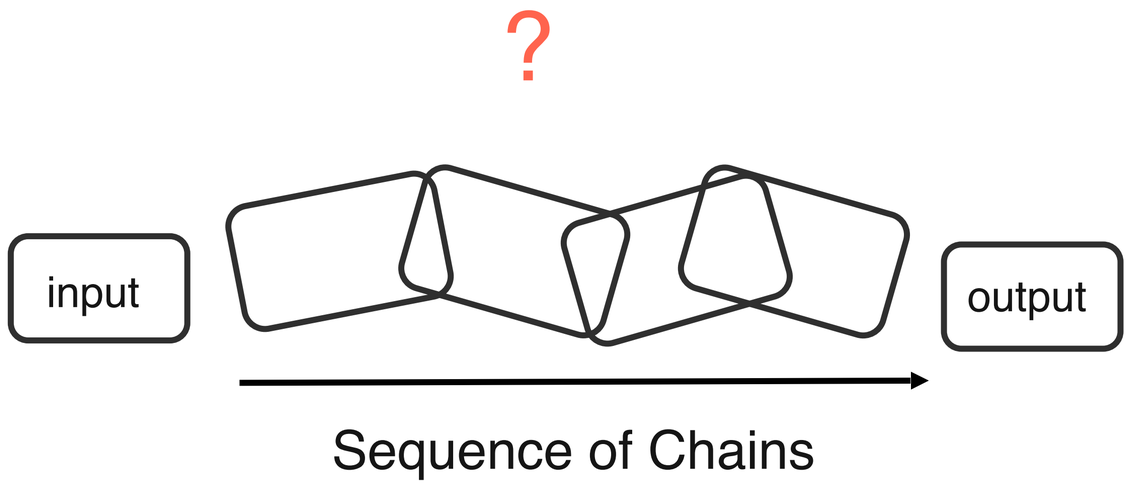
- Sequence of calls to an LLM chained together



Basics of Langchain

LLMChain & Sequential Chains

- **Chains:** prompt template + Model + output parsing
- Sequence of calls to an LLM chained together
- How to build these chains?



Basics of Langchain

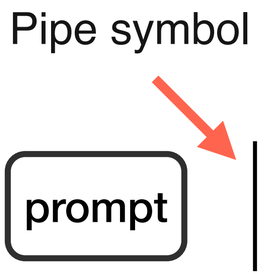
LCEL - LangChain Expression Language

LCEL - designed to build sequence of calls (to LLMs or any other component)

Basics of Langchain

LCEL - LangChain Expression Language

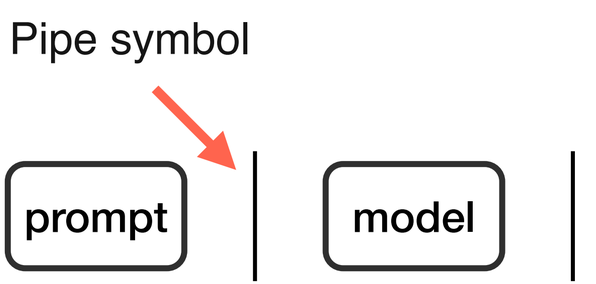
- **LCEL** - designed to build sequence of calls (to LLMs or any other component)
- Pipe syntax: prompt **|**



Basics of Langchain

LCEL - LangChain Expression Language

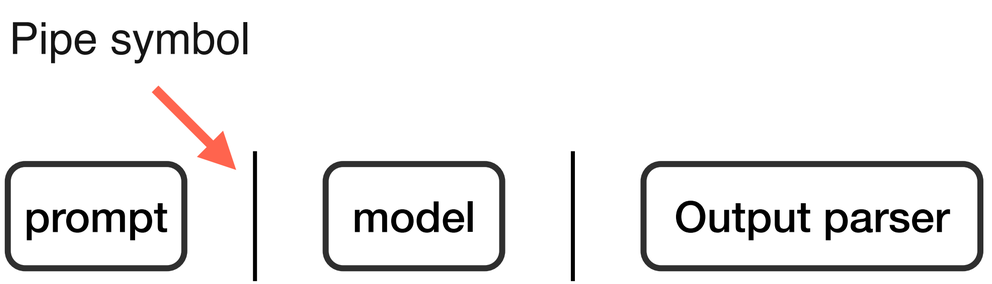
- **LCEL** - designed to build sequence of calls (to LLMs or any other component)
- Pipe syntax: prompt **|** model **|**



Basics of Langchain

LCEL - LangChain Expression Language

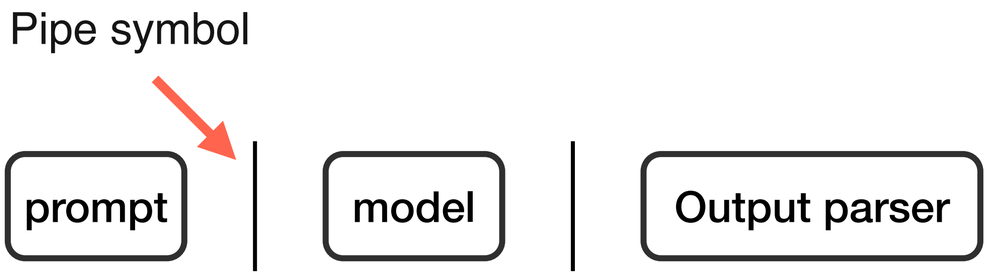
- **LCEL** - designed to build sequence of calls (to LLMs or any other component)
- Pipe syntax: prompt **|** model **|** output parser



Basics of Langchain

LCEL - LangChain Expression Language

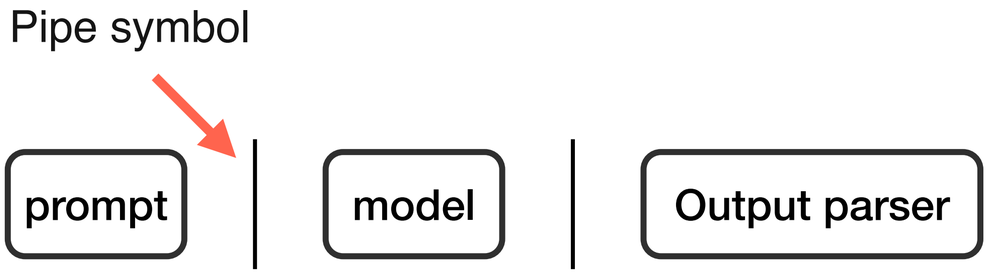
- **LCEL** - designed to build sequence of calls (to LLMs or any other component)
- Pipe syntax: prompt **|** model **|** output parser
- Allows you to **build complex chain pipelines** with a simple standard interface



Basics of Langchain

LCEL - LangChain Expression Language

- **LCEL** - designed to build sequence of calls (to LLMs or any other component)
- Pipe syntax: prompt **|** model **|** output parser
- Allows you to **build complex chain pipelines** with a simple standard interface

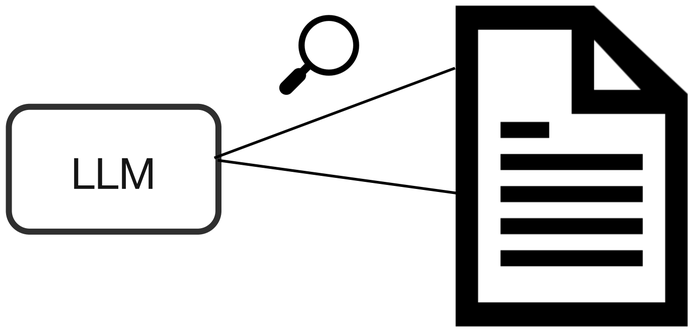


Notebook demo - Building Chains with LangChain

Langchain for LLM App Development

Langchain with Documents

- LLMs have a limited context length



Langchain for LLM App Development

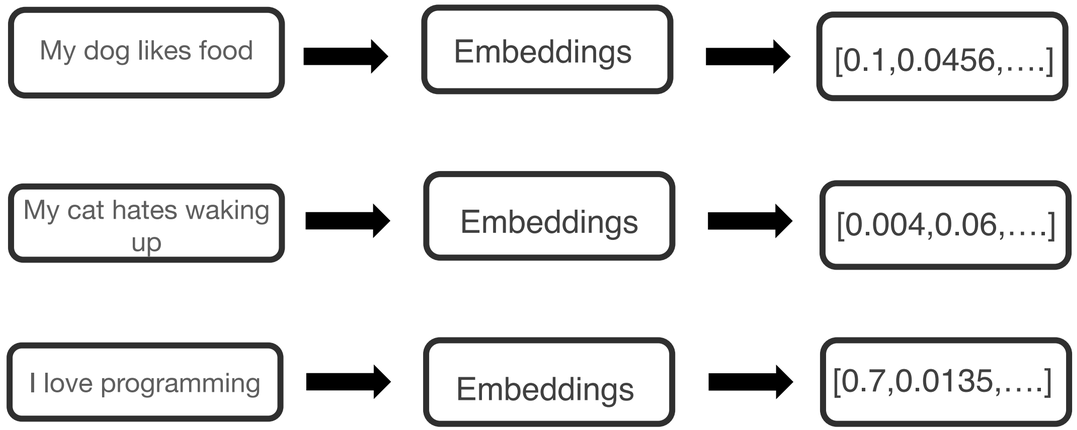
Langchain with Documents

- LLMs have a limited context length
- **Embeddings**: capture content and meaning

Langchain for LLM App Development

Langchain with Documents

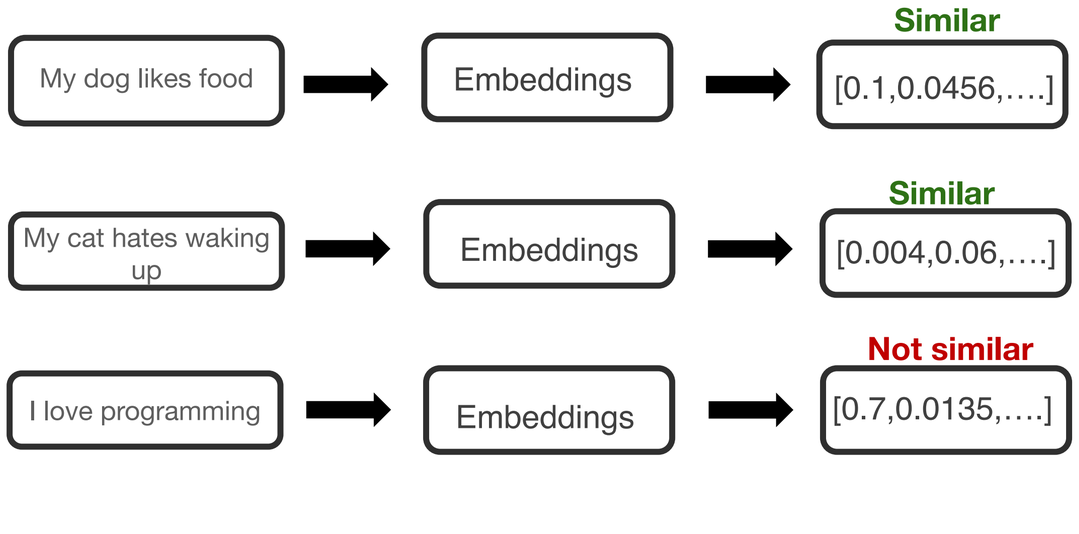
- LLMs have a limited context length
- **Embeddings**: capture content and meaning



Langchain for LLM App Development

Langchain with Documents

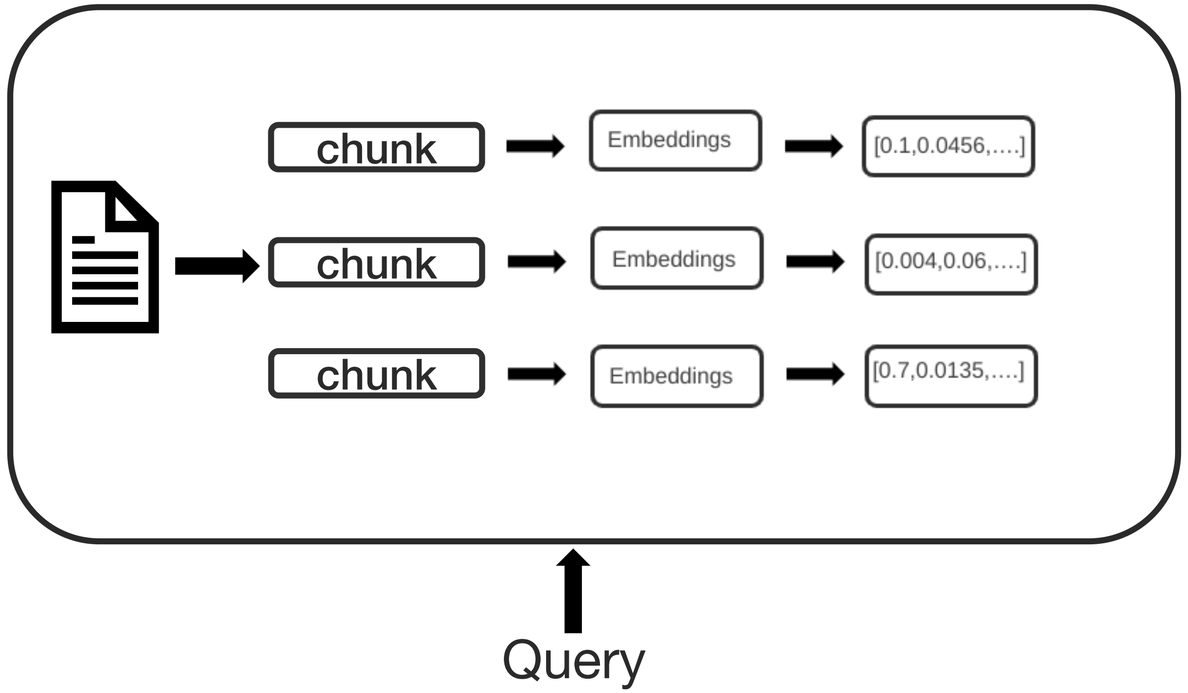
- LLMs have a limited context length
- **Embeddings**: capture content and meaning



Langchain for LLM App Development

Langchain with Documents

- LLMs have a limited context length
- Embeddings: capture content and meaning
- **Vector DBs**



Langchain for LLM App Development

Langchain with Documents

- LLMs have a limited context length
- Embeddings: capture content and meaning
- Vector DBs

Chunking methods for large docs

- Map_reduce
- Refine
- Map_rerank

Langchain for LLM App Development

Langchain with Documents

- LLMs have a limited context length
- Embeddings: capture content and meaning
- Vector DBs

Chunking methods for large docs

- Map_reduce
- Refine
- Map_rerank

Notebook demo: Chat with Data using LangChain