NOVA IMS
Information Management School

7

LARGE LANGUAGE MODELS

Business Cases with Data Science

Instituto Superior de Estatística e Gestão da Informação
Universidade Nova de Lisboa

Acreditações e Certificações

NOVA IMS Information Management School

Summary

1. Introduction
2. Generative AI
3. Prompt engineering
4. Retrieval-augmented generation
5. Parameters tuning
6. ChatGPT: Getting an API Key
7. Azure Open AI

NOVA IMS Information Management School

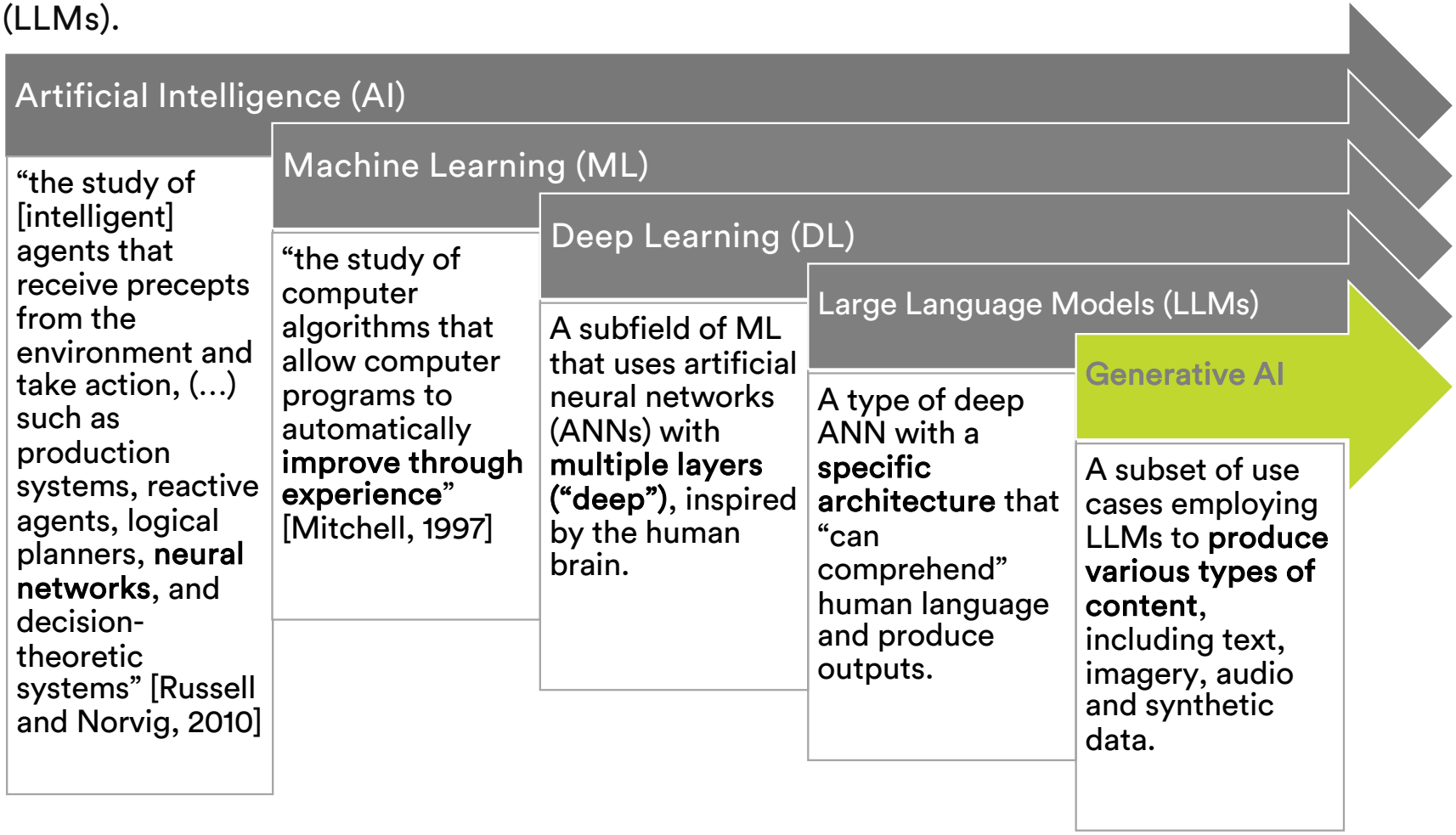
7.1

Introduction

From Neural Networks to Large Language Models

Evolution of Machine Learning to Generative AI

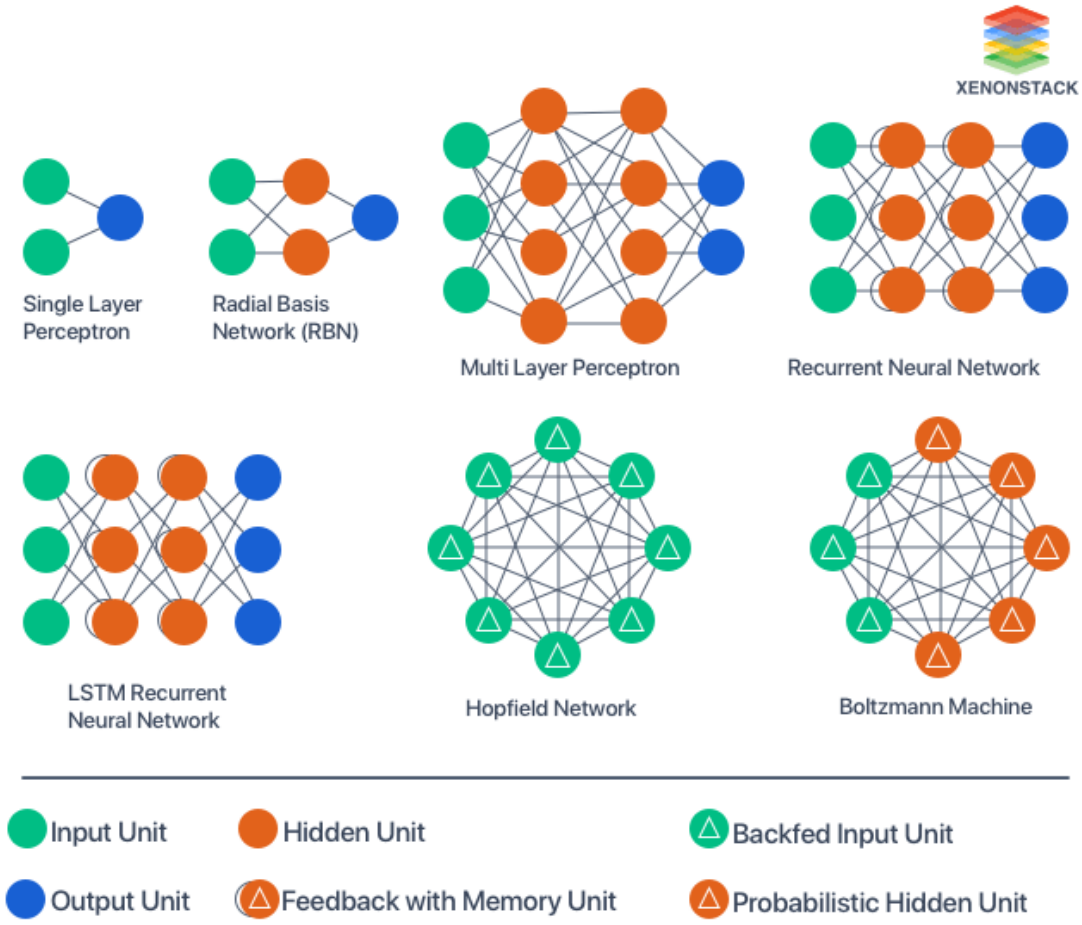
The foundations that led to the development of Large Language Models (LLMs).



Generative AI context

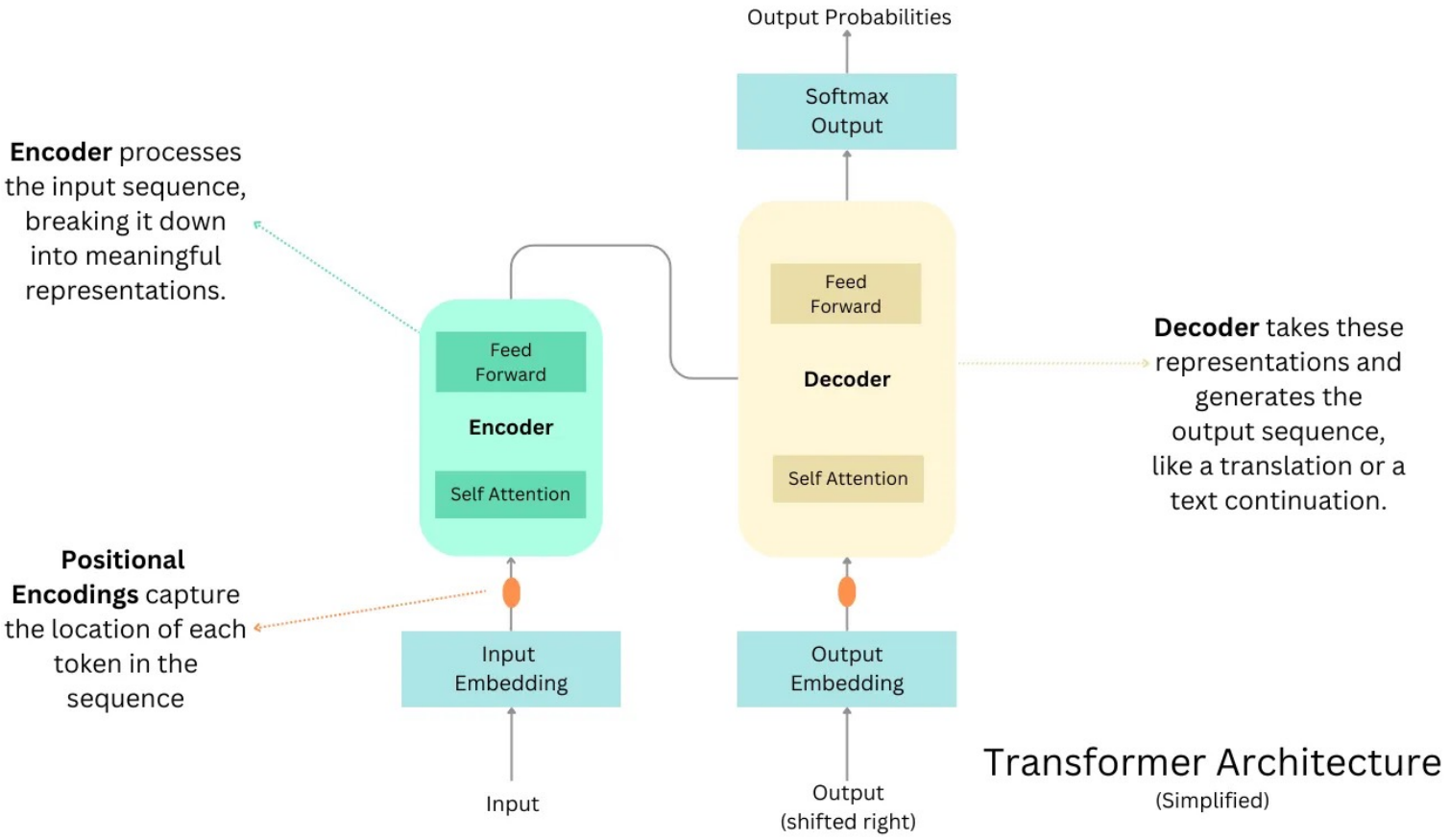
WHAT is Generative AI?	A subset of artificial intelligence focused on the **ability of machines to create outputs across various modalities** *(e.g., text, images, audio, code, voice, video)*
HOW does it work?	It uses **LLMs**, such as OpenAI's GPT, trained on massive amounts of data to understand human communication and natural language
WHY now?	Innovations in hardware, cloud-native stack, software engineering, machine learning, and deep learning allowed the **creation of Generative AI** solutions
WHO is involved?	Technology leaders and start-ups are developing user-facing applications on these underlying models, with **apps such as ChatGPT, and Dall-E** rising in popularity

The roadmap to LLMs: multiple architectures



source: https://www.tomorrow.bio/post/nodes-that-know-how-neural-network-architecture-learns-2023-06-4669541450-ai

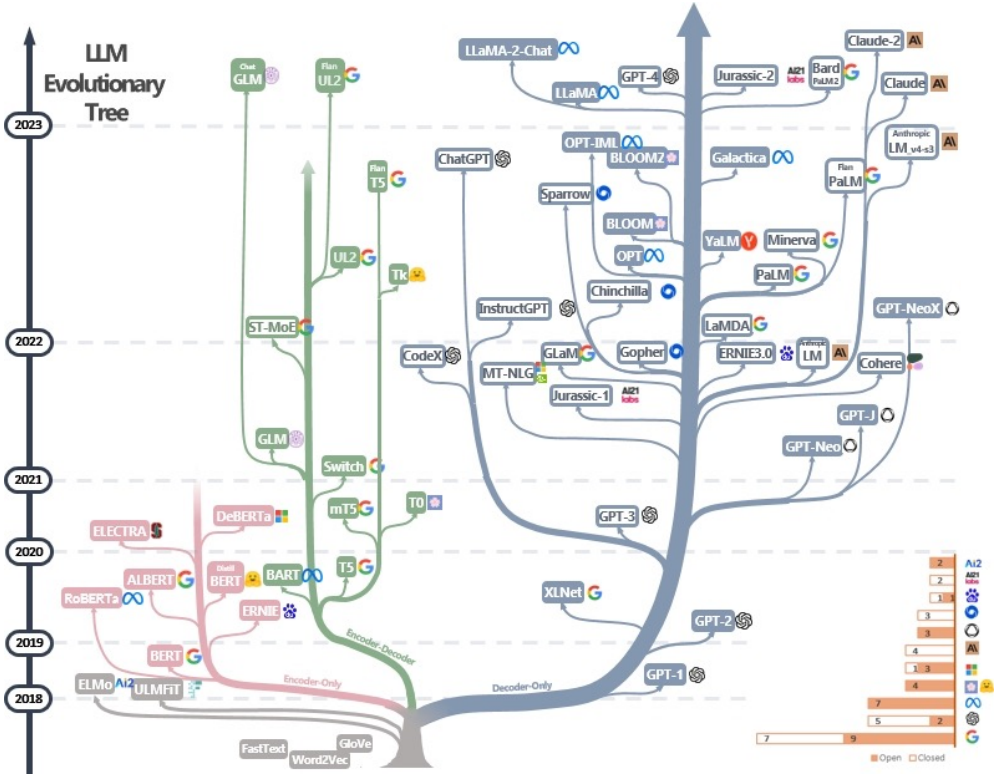
Transformer: the game changer



source: https://medium.com/@tech-gumptions/transformer-architecture-simplified-3fb501d461c8

LLM model types and evolution (1/3)

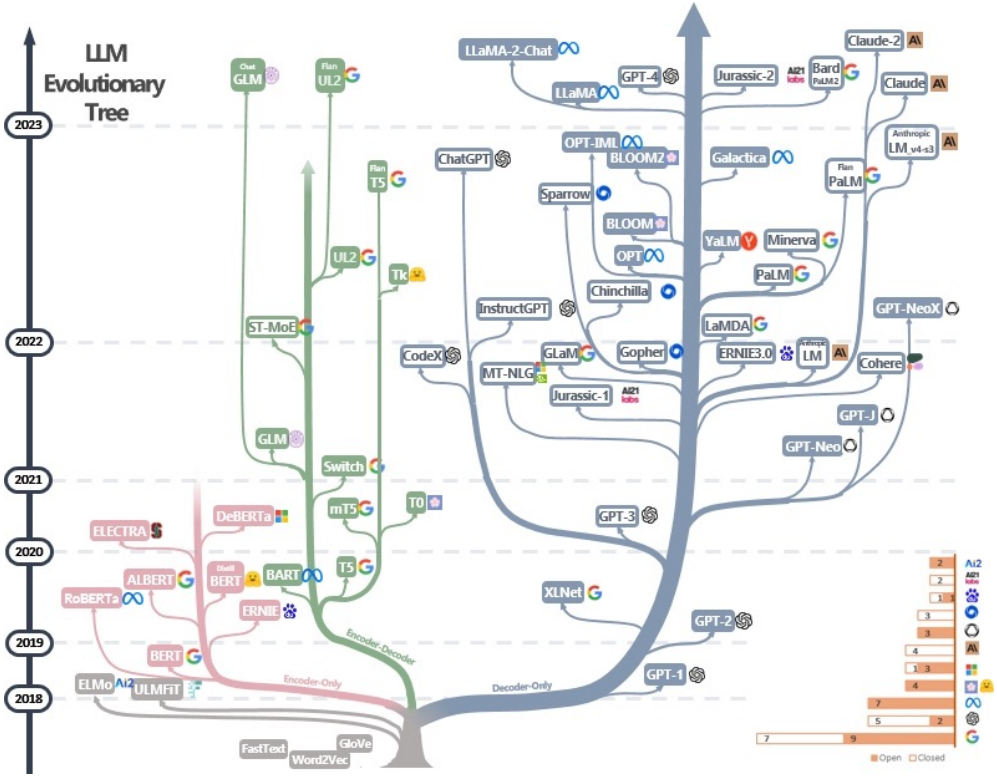
Encoder-decoder: Architecture with two parts. The encoder processes the input (e.g., English sentence) and compresses it in a context. The decoder takes the context and produces an output (e.g., a translation to French)



Source: https://github.com/Mooler0410

LLM model types and evolution (2/3)

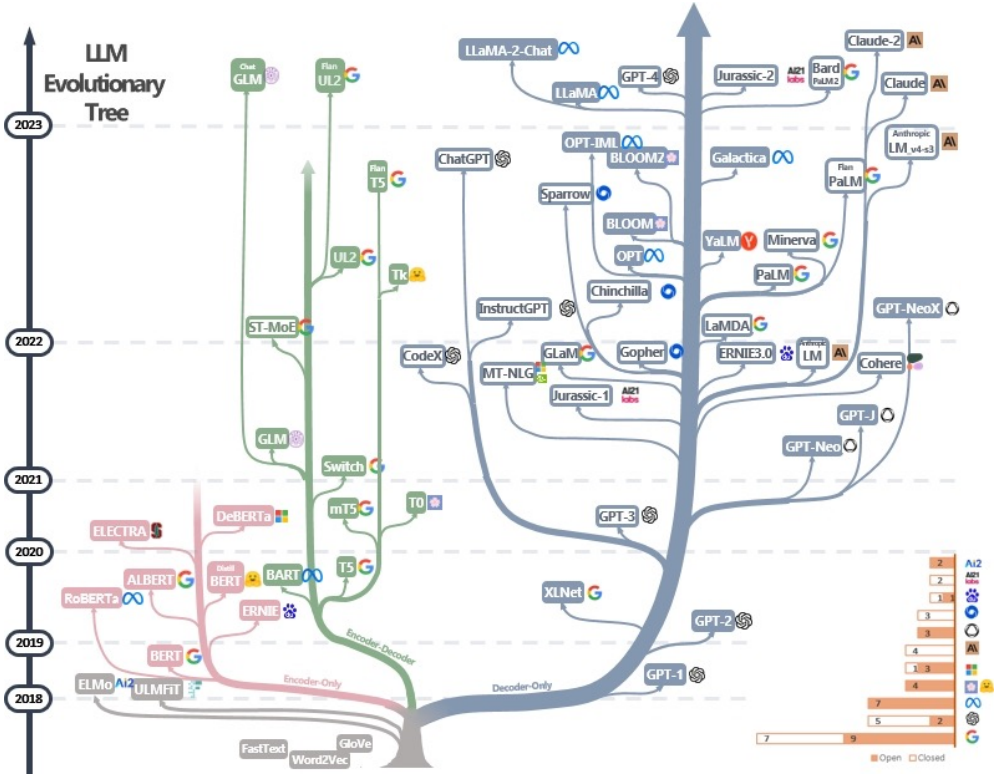
Encoder-only: Architecture with just the encoder. It processes the input and produces a direct output. Useful for tasks where you do not need a transformation into another ”type” or “form” (e.g., classification tasks)



Source: https://github.com/Mooler0410

LLM model types and evolution (3/3)

Decoder-only: Architecture with only the encoder. It starts with basic information and expands or generates output based on it. This what models like GPT use (generate stories, answers, etc.)



Source: https://github.com/Mooler0410

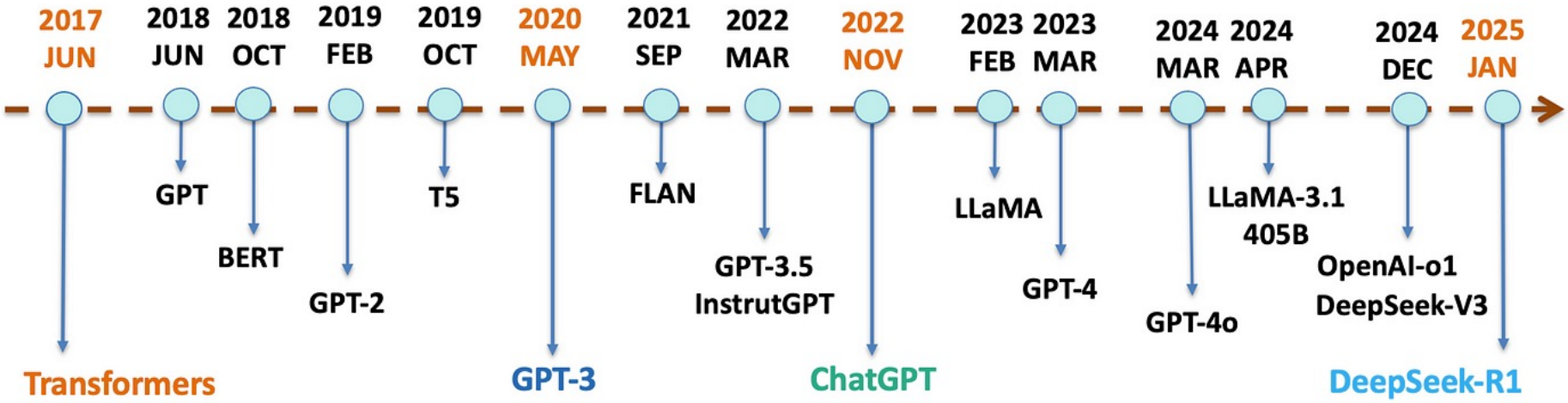
NOVA IMS Information Management School

7.2

Generative AI

LLMs as enablers of the Generative AI

The evolution of LLMs



source: medium.com

LLMs are undergoing rapid evolution, making them applicable across diverse contexts and realities...

- Thousands of open-source LLMs are available
- Hundreds of lighter versions of models, that require less powerful hardware, facilitating broader accessibility and ease of implementation
- Dozens of fine-tuned versions of the main models

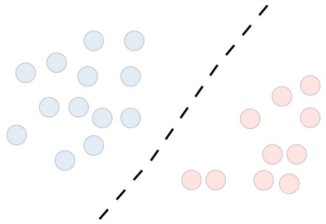
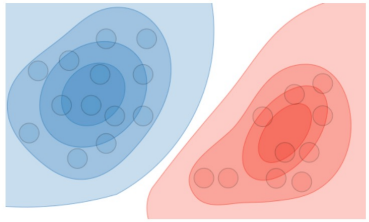
Types of Deep Learning Models

Discriminative

- Used to classify or predict
- Typically trained with labeled data
- Learns the relationship between the features of the data points and the labels

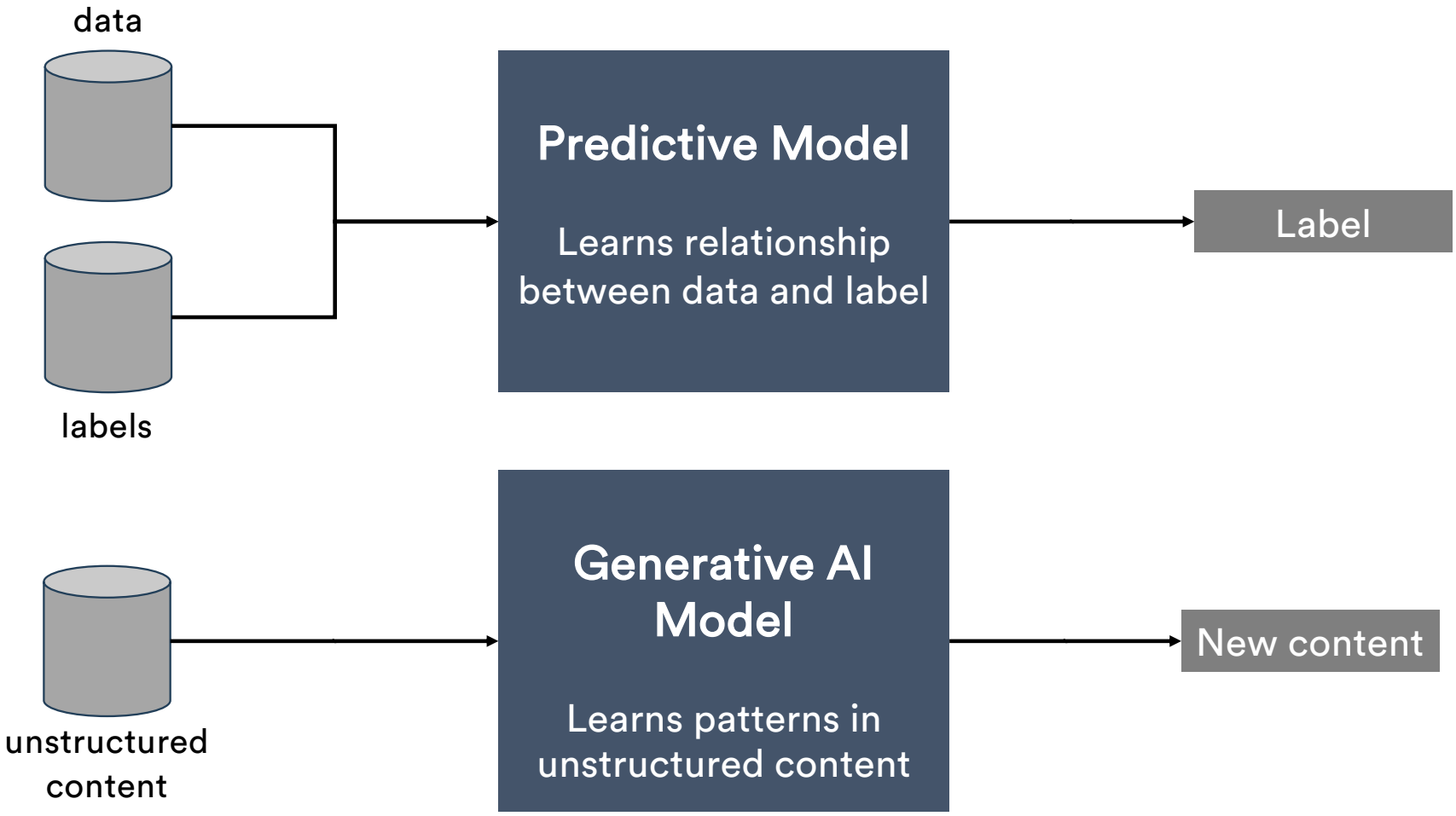
Generative

- Generates data points similar to the ones it was trained with
- Understand the data distribution and how likely a given example is
- Predict next “word” in a sentence

	Discriminative model	Generative model
Goal	Directly estimate $P(y\|x)$	Estimate $P(x\|y)$ to then deduce $P(y\|x)$
What's learned	Decision boundary	Probability distributions of the data
Illustration		
Examples	Regressions, SVMs	GDA, Naive Bayes

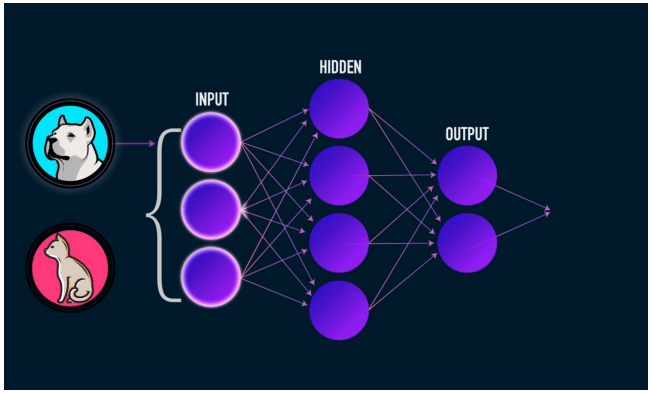
source: standford.edu

Predictive vs Generative Models



LLMs sizes

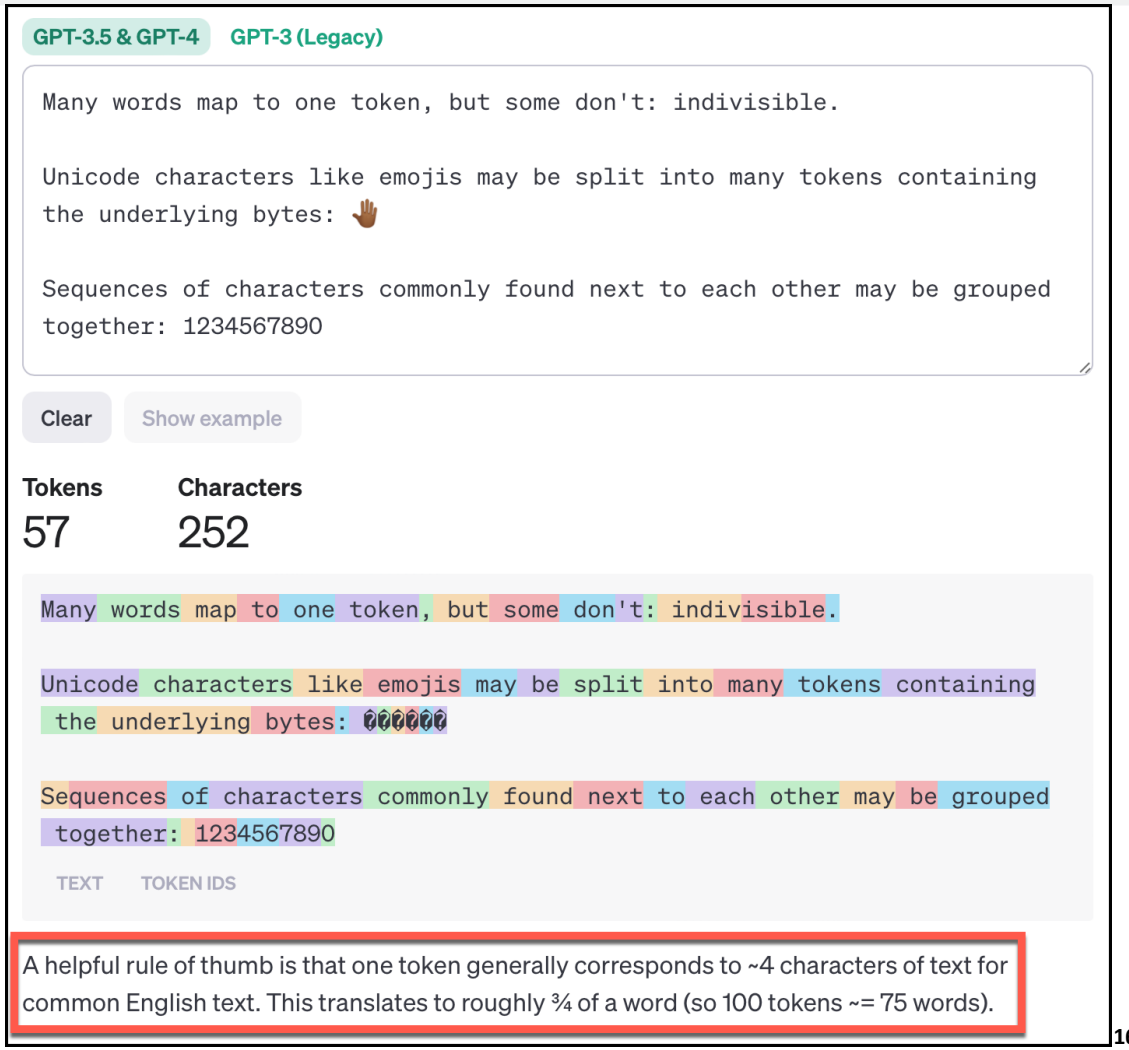
- LLMs are measured by the number of **parameters** in the model and the number of **tokens** they were trained on
- **Parameters** are the weights and biases connecting nodes in a neural network
- **Tokens** are how LLMs break up sentences into “pieces”. These can be multiple words, single words, or parts of words
- Generally, the more parameters a model has and the more tokens it was trained, the better it performs



Source: https://towardsdatascience.com/visualizing-artificial-neural-networks-anns-with-just-one-line-of-code-b4233607209e

Text tokenization (1/3)

- Foundational step in Natural Language Processing (NLP) processing, including in LLMs like GPT-4 and Llama-2
- Involves breaking text into smaller chunks (tokens), which can be as short as one character or as long as one word (in some cases multiple words)
- Tokens are then processed and used as input for Machine Learning models



Source: https://platform.openai.com/tokenizer

Text tokenization (2/3)

GPT-3.5 & GPT-4 GPT-3 (Legacy)

Many words map to one token, but some don't: indivisible.

Unicode characters like emojis may be split into many tokens containing the underlying bytes: 🤚

Sequences of characters commonly found next to each other may be grouped together: 1234567890

Clear Show example

Tokens 57 **Characters** 252

Many words map to one token, but some don't: indivisible.

Unicode characters like emojis may be split into many tokens containing the underlying bytes: ������

Sequences of characters commonly found next to each other may be grouped together: 1234567890

TEXT TOKEN IDS

A helpful rule of thumb is that one token generally corresponds to ~4 characters of text for common English text. This translates to roughly ¾ of a word (so 100 tokens ~= 75 words).

GPT-3.5 & GPT-4 GPT-3 (Legacy)

Many words map to one token, but some don't: indivisible.

Unicode characters like emojis may be split into many tokens containing the underlying bytes: 🤚

Sequences of characters commonly found next to each other may be grouped together: 1234567890

Clear Show example

Tokens 57 **Characters** 252

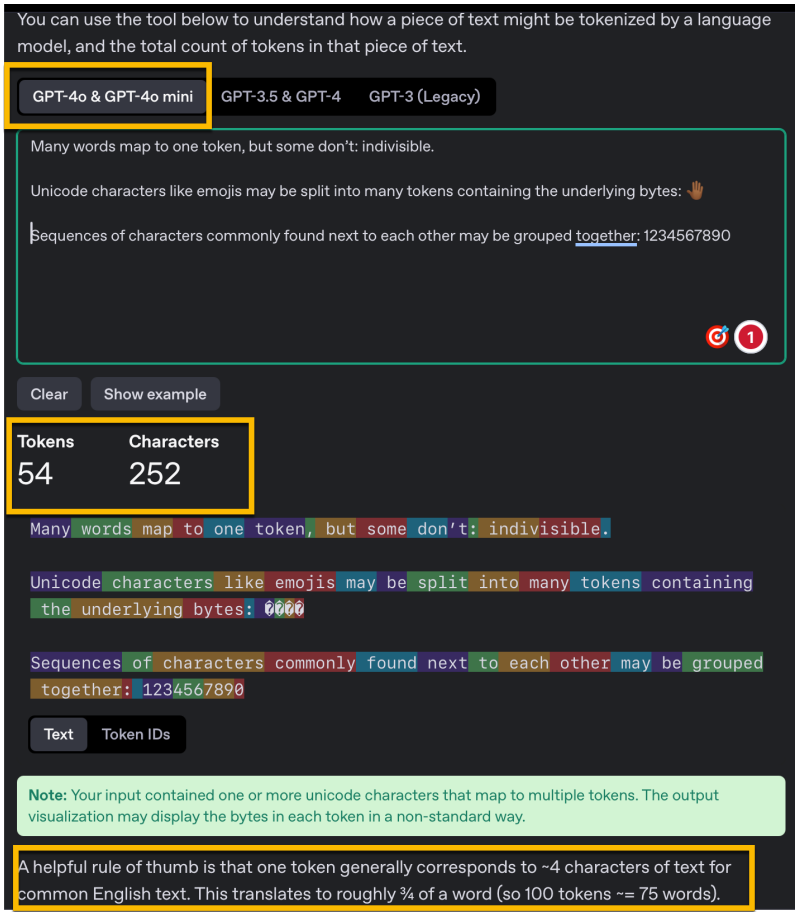
[8607, 4339, 2472, 311, 832, 4037, 11, 719, 1063, 1541, 956, 25, 3687, 23936, 382, 35020, 5885, 1093, 100166, 1253, 387, 6859, 1139, 1690, 11460, 8649, 279, 16940, 5943, 25, 11410, 97, 248, 9468, 237, 122, 271, 1542, 45045, 315, 5885, 17037, 1766, 1828, 311, 1855, 1023, 1253, 387, 41141, 3871, 25, 220, 4513, 10961, 16474, 15]

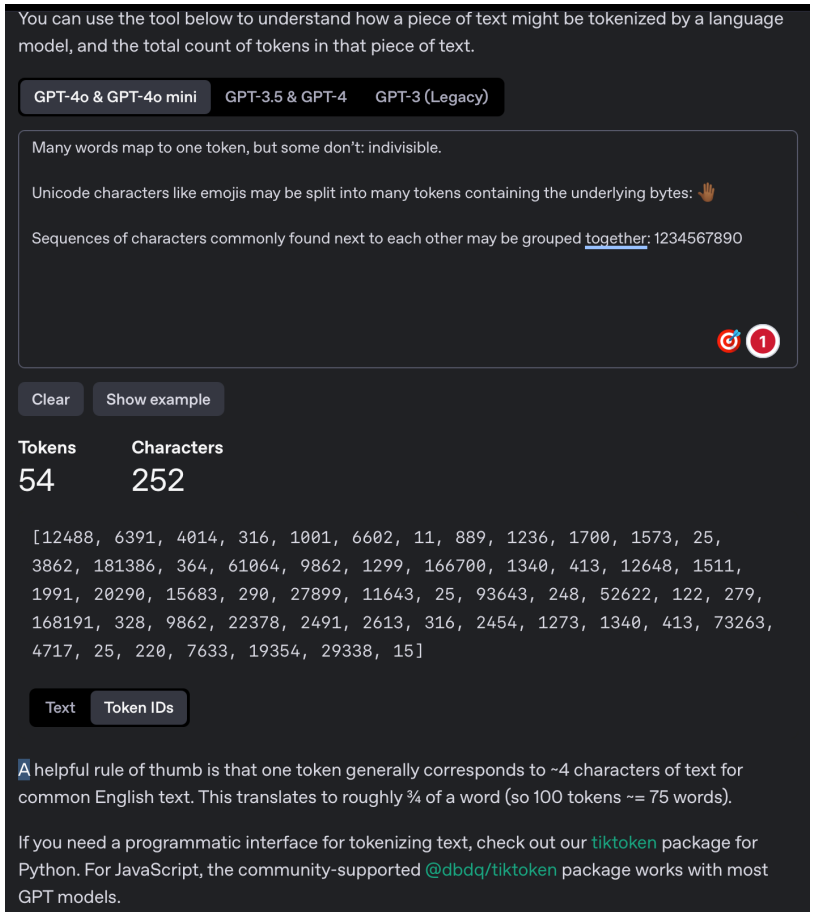
TEXT TOKEN IDS

A helpful rule of thumb is that one token generally corresponds to ~4 characters of text for common English text. This translates to roughly ¾ of a word (so 100 tokens ~= 75 words).

Source: https://platform.openai.com/tokenizer

Text tokenization (3/3)





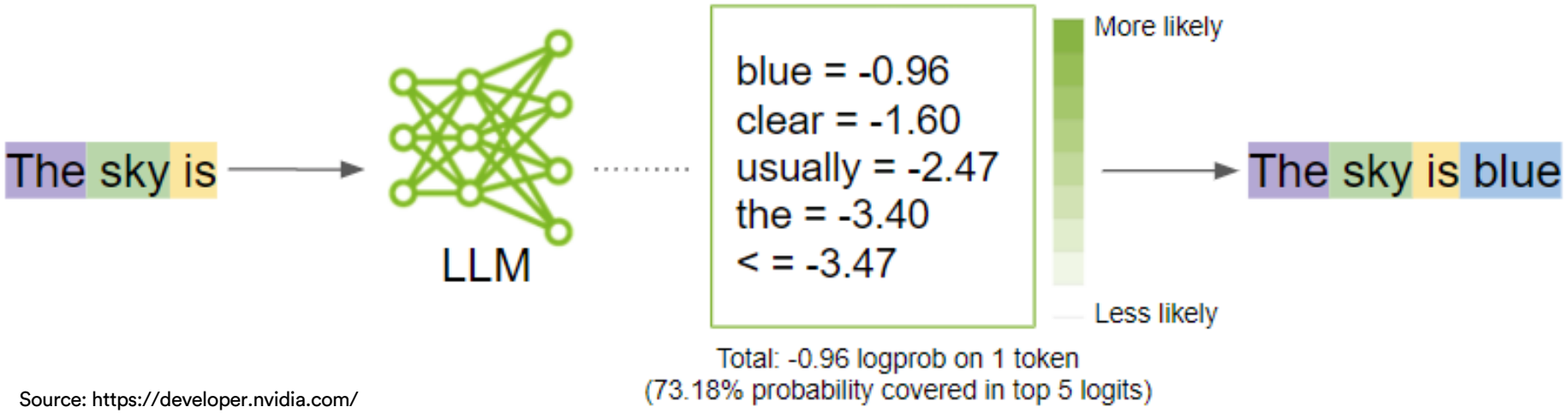
Source: https://platform.openai.com/tokenizer

LLMs are autoregressive language models

LLMs take an input text and repeatedly predict the next token based on the context.

$$P(token_k|token_{context}) = \frac{\exp(logit_k)}{\sigma_j exp(logit_j)}$$

This is the probability of $token_k$ given the context from previous tokens ($token_1$ to $token_{k-1}$) and $logit_k$ is the output of the neural network.



Source: https://developer.nvidia.com/

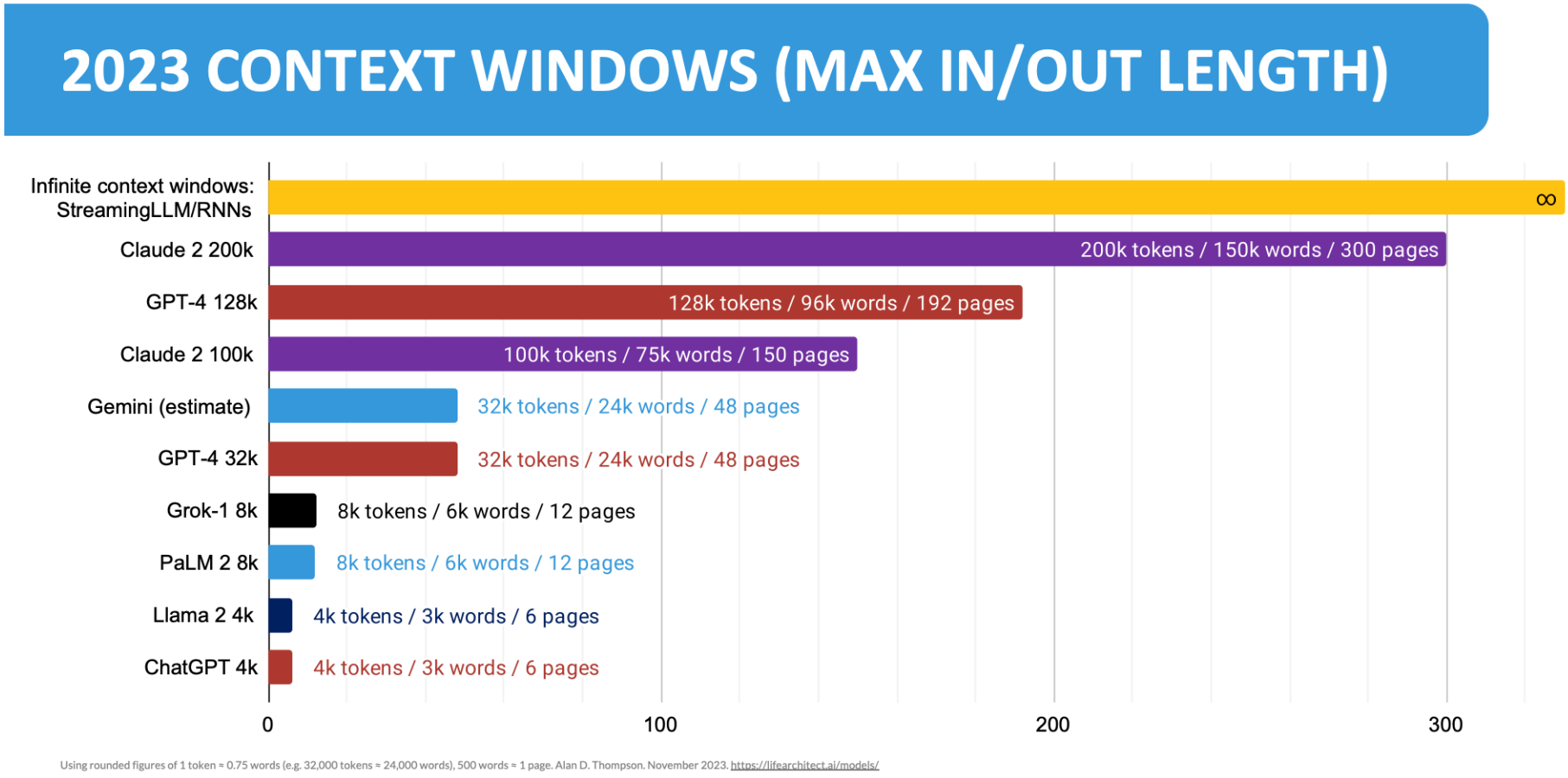
LLMs context window

The context window (or “context length”) of an LLM is the amount of text, in **tokens**, that the model can consider or “remember” at any one time. A larger context window enables an AI model to **process longer inputs** and **incorporate more information into each output**.

Increasing an LLM’s context window size translates to increased accuracy, fewer hallucinations, more coherent model responses, longer conversations, and an improved ability to analyze longer sequences of data.

However, increasing context length is not without tradeoffs: it often entails increased computational power requirements, which in turn increases costs.

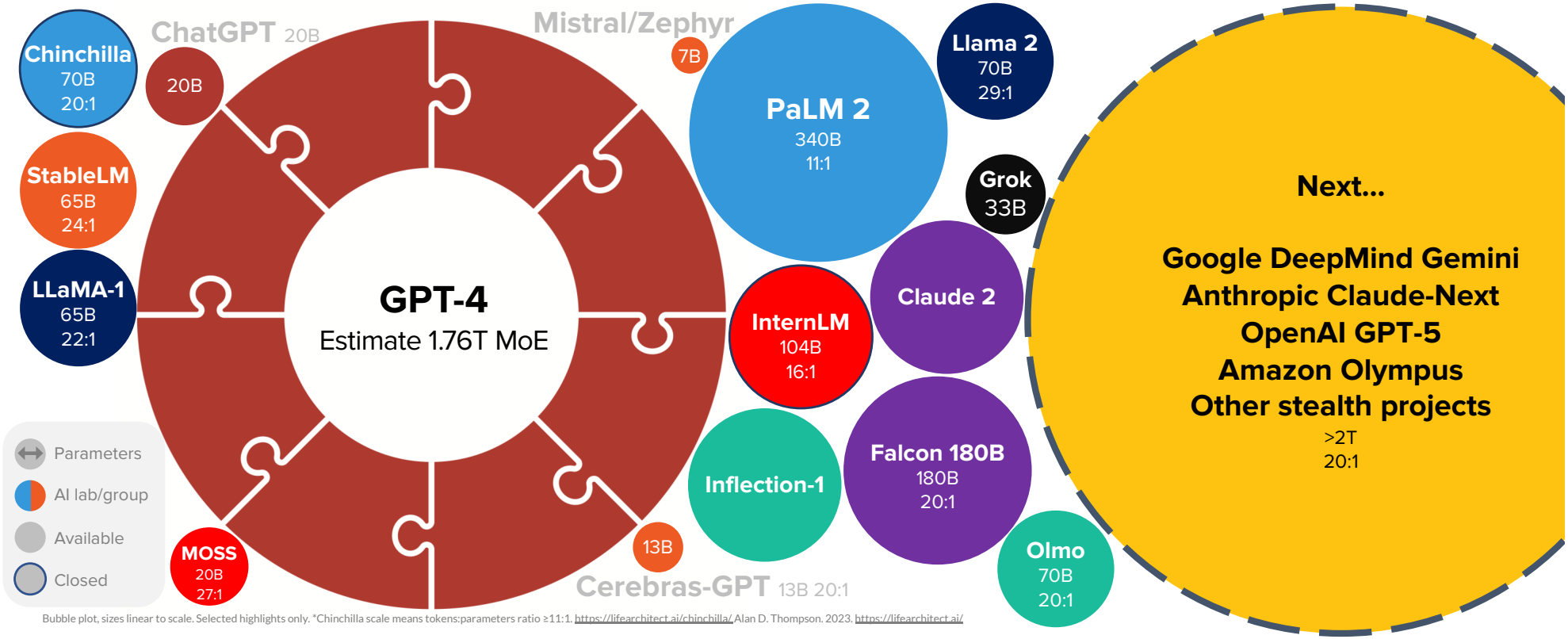
LLMs context size



NOVA IMS Information Management School

2023-2024 OPTIMAL LANGUAGE MODELS

NOV/ 2023



Bubble plot, sizes linear to scale. Selected highlights only. *Chinchilla scale means tokens:parameters ratio ≥11:1. https://lifearchitect.ai/chinchilla/ Alan D. Thompson. 2023. https://lifearchitect.ai/

LifeArchitect.ai/models

NOVA IMS Information Management School

Generative LLMs

AI Titans Face Off

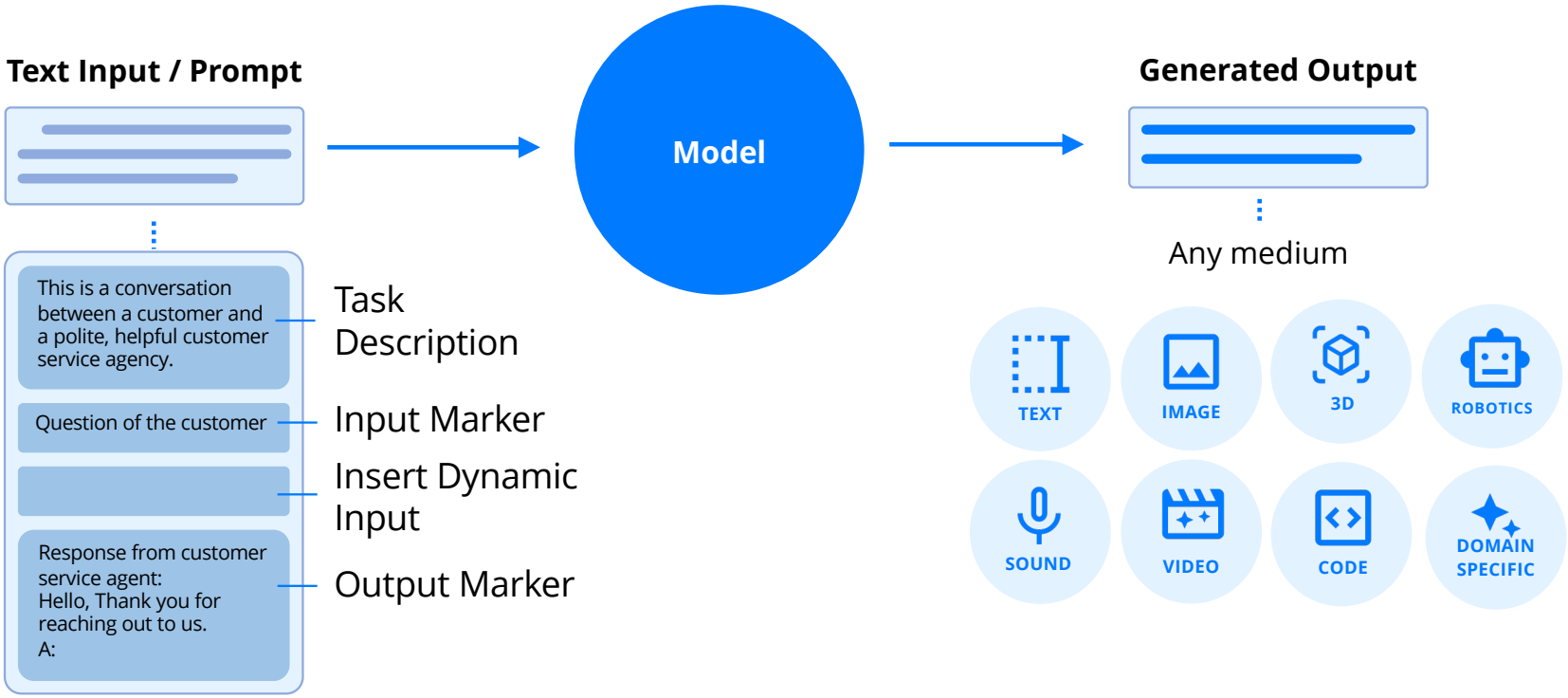
	GPT-4o	Gemini 1.5 001	Llama 3.1 7B	Mistral 7B	Claude 3.5	DeepSeek V3
Model Overview						
Country	USA	USA	USA	France	USA	China
Developer	OpenAI	Google	Meta	Mistral AI	Anthropic	DeepSeek Inc.
Release Date (**model released**)	2023	2024	2024	2023	2023	2023
Capabilities and Features						
Supported Modalities	Text, Image	Text, Image, Video, Voice	Text	Text	Text, Image	Text
Capabilities	Excels in coding, complex tasks, versatile applications	Multimodal with advanced reasoning, data processing	Strong in language math with improved generation	Specialized in language tasks with open-source efficiency	Safe, nuanced text generation ethical AI alignment	Specialized in scientific research and data analysis
Training and Updates						
Training Data	Extensive web data	Proprietary data	Large web data	Open web resources	Open web resources	Proprietary datasets
Ethical Considerations	Bias, Misinformation	Bias, Privacy concerns	Bias, Privacy concerns	Bias, Privacy concerns	Strong focus on safety	Bias, Privacy concerns
Knowledge Updates	10/2023	11/2023	12/2023	Unknown	08/2023	Unknown
Performance Metrics (Tokens)						
Number of Tokens Supported	128K	1M	128K	32K	200K	128K
Tokens That Can Be Generated	16.4K	8K	2K	8K	4K	8K
Accessibility and Licensing						
Open Source	No	No	Yes	Yes	No	Yes
API Providers	OpenAI Azure OpenAI	Google AI Studio Vertex AI	Azure AI AWS Bedrock Google AI Studio Vertex AI IBM watsonx Deepinfra	Azure AI AWS Bedrock Google AI Studio Vertex AI Snowflake Cortex	Anthropic, AWS Bedrock Google AI Studio Vertex AI	DeepSeek HuggingFace
Pricing and Licensing						
Cost/Licensing	Subscription, Pay-per-use	Subscription, Pay-per-use	Open-source, free to use	Open-source, free to use	Open-source, free to use	Details Unclear
Input Cost (per Million token)	$2.50	$0.13	$0.23	$0.25	$15.00	$0.14
Output Cost (per Million token)	$10.00	$0.38	$0.40	$0.25	$75.00	$0.28

Ghazi MEJAT
Daily AI content

Use the right AI for the right task

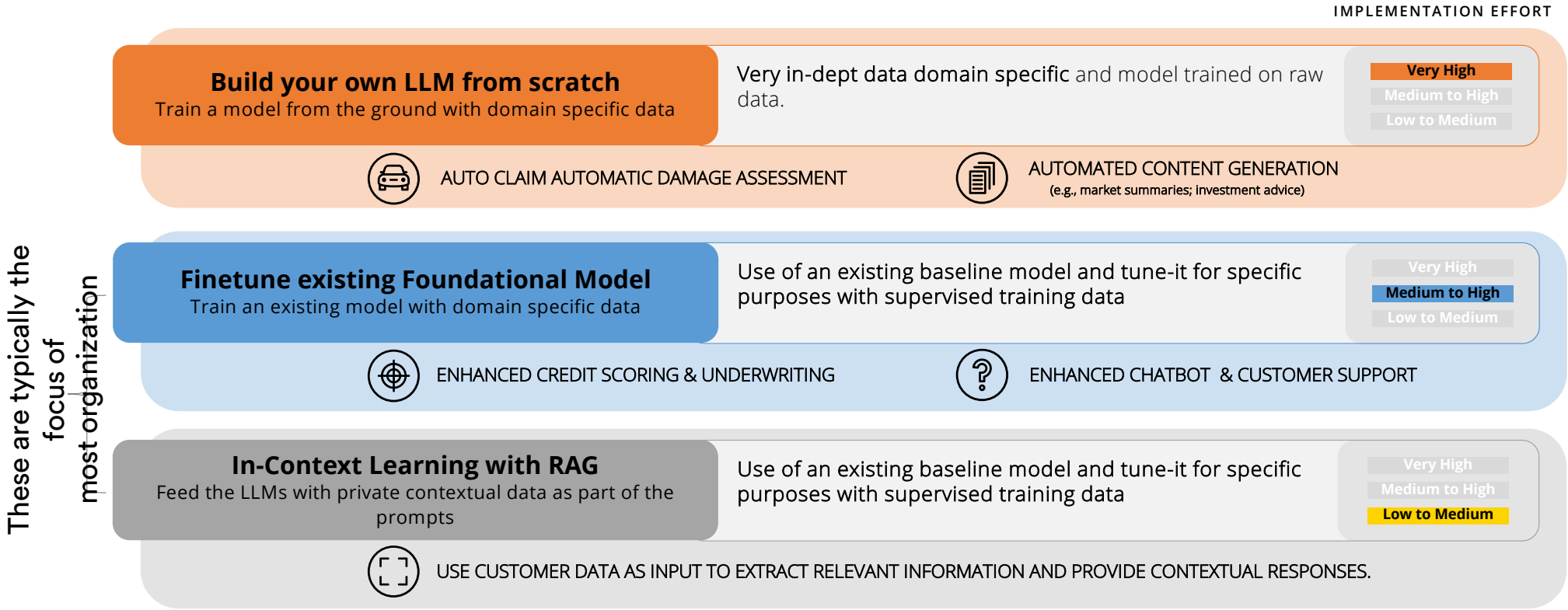
REPOST

How Generative AI applications work?



How models adapt to new domains?

Depending on the use case, models may need to adapt to new domains or be exposed to new data. The implementation efforts (time, cost and compute effort) are completely different and may impact the decision.



NOVA IMS Information Management School

7.3

Prompt engineering

Designing inputs to produce optimal outputs

Prompt engineering

- It is the process of transmitting instructions to LLMs
- A prompt is a Natural Language text describing the task the model should perform
- A prompt can be:
  - A query (e.g., “What is the Theory of relativity?”)
  - A command (e.g., “Write a post for NOVA IMS Instagram”)
  - A role assignment (e.g., Act as a Portuguese native speaker)
  - A statement on feedback (e.g., “answer in a formal way”)
  - A statement on context (e.g., providing a text to ask for a summarization)
- Creating prompts requires creativity and attention to detail
- Prompts should be as clear and specific as possible

Why being specific is important

https://www.youtube.com/watch?v=cDA3_5982h8&t=13s

Types of operations

Prompt Engineering Guide

There are 3 main types of prompt operations that can be performed:

REDUCTIVE PROMPT OPERATIONS

Takes a large amount of text and makes it shorter

- Summarizations (executive summary, translate text to bullets, etc.)
- Extractions (retrieve data from a text, example: extract dates, names, etc.)
- Characterizations (describe a specific text)
- Analyzing (find text patterns)
- Critic (provide contextual feedback on a text, as sentiment analysis.)

TRANSFORMATION PROMPT OPERATIONS

Take a given input and transform it

- Reformatting (change the text output format, example: JSON to YAML or to XML)
- Language translations (translate text from English to Portuguese)
- Code Languages Translations (example convert COBOL to Python)
- Restructuring (reorganize code or make it more performance efficient)
- Tone Modification (change the communication output to be more formal)

GENERATIVE OPERATIONS

Create new, unseen content

- Planning (draft a vacations plan given a list of locations)
- Drafting (write an introduction for a letter or a draft plot for a fiction book
- Brainstorm (imagine a list of possibilities given a particular use case)
- Expansion (generate more content given a short text input)

Basics of prompt engineering (1/3)

Be specific and clear:

- Use delimiters (“, -, #, <, >) to split the different sections of your prompt (e.g., defining steps or framing a user message)
- Define the output format. If JSON or XML, give an example if possible
- Be clear in assumptions and conditions. For example, the return value should be from a list of three possible values
- Give a few input and output examples. This is called few-shot prompting

Basics of prompt engineering (2/3)

Push the model to **think about** the answer:

- Give step-by-step instructions to force it to go through all the steps
- If the previous approach is not viable, break the task into smaller sequential prompts

Basics of prompt engineering (3/3)

"LLMs are 90% correct and 100% confident"

Be aware of hallucinations:

- Ask the model to link the answer to relevant information from the context, then answer the question based on the data
- If the model uses the Internet, ask for URL references
- If running from the API, reduce "temperature" to a low value

Zero-shot prompting

Q: Write an answer to the following customer complaint with at most 300 characters.
###
Complaint: This is one of the worst companies I've ever done business with. They don't answer the phones and don't care about the customer.

A: We're sorry to hear about your experience. We strive to provide excellent customer service and regret any inconvenience. Please email us at support@example.com with your contact details, and we'll ensure prompt assistance.

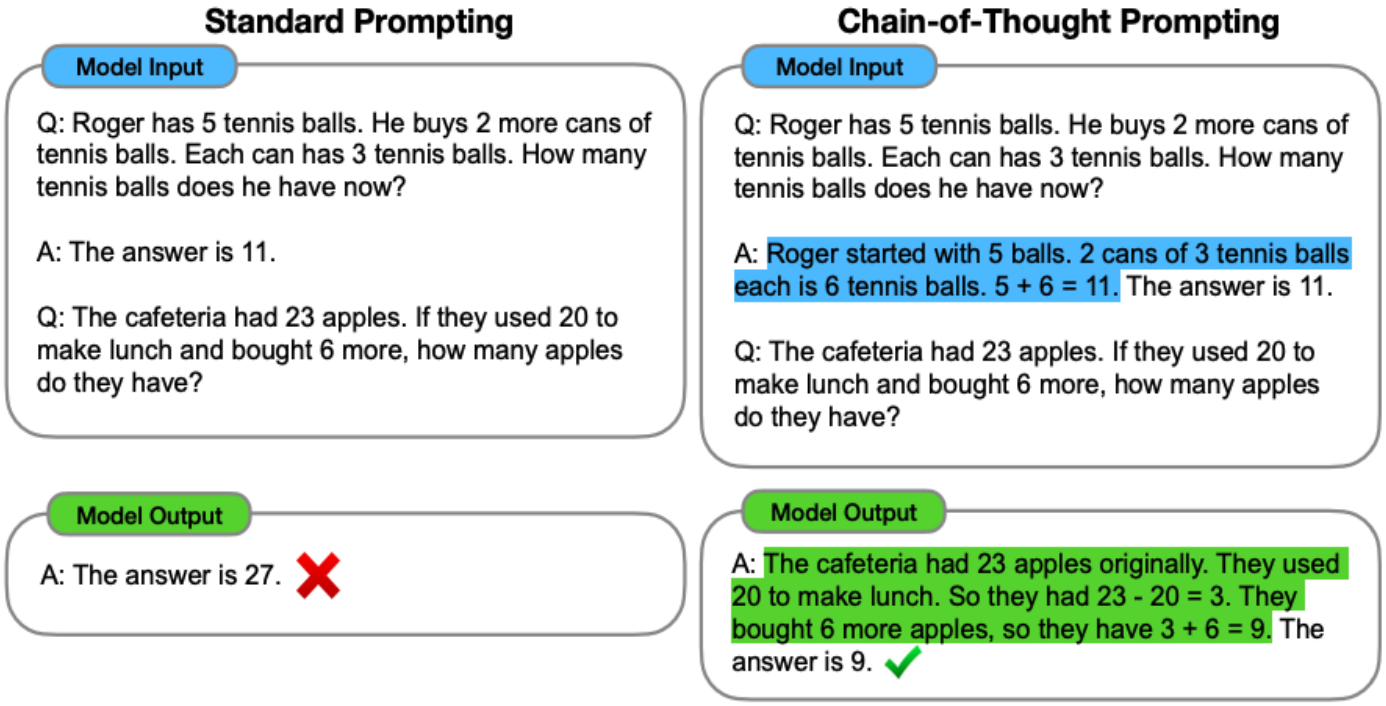
Few-shot prompting

Q: Considering the examples below, write an answer to the following customer complaint with at most 300 characters:
###
Complaint: This is one of the worst companies I've ever done business with. They don't answer the phones and don't care about the customer.
###
Example 1: I'm sorry to hear that [product name] did not meet your expectations. We strive to ensure every customer is happy with their purchase, and it's clear we have not lived up to this promise in your case. To better assist you, could you please share specific details about how the product did not meet your expectations?
Example 2: I'm so sorry for the delay in getting back to you. I understand your frustration. Your email needed and deserved a timely response. Please contact me directly if you experience any further problems on the site.

A: We're **truly** sorry to hear about your experience **and the frustration it has caused**. We **value our customers deeply** and would like to resolve this **promptly**. Please email us **directly** at help@ourcompany.com so we can address your concerns **immediately**.

Chain-of-thought (CoT) prompting

CoT prompting enables complex reasoning capabilities through intermediate reasoning steps. Can be combined with few-shot prompting to deliver better results.



source: Wei et al. (2022), https://arxiv.org/abs/2201.11903

Using prompt schemas

```
import openai

# Define user inputs and campaign details
product_name = "Eco-Friendly Water Bottle"
target_audience = "environmentally conscious consumers"
campaign_theme = "sustainability and durability"
key_features = ["made from 100% recycled materials",
                "keeps drinks cold for 24 hours"]

# Build a detailed prompt incorporating context, persona, and model role
prompt = f"""
Create a script for a marketing campaign for a product \
called '{product_name}'.
Target audience: {target_audience}.
Campaign theme: {campaign_theme}.
Key features of the product:
- {key_features[0]}
- {key_features[1]}

The script should be engaging and inspire the audience \
to support sustainability by choosing this product. Use \
a friendly and persuasive tone, suitable for social media \
platforms. Include a call to action at the end of the script.

Example of a good script:
"Join us in making the earth greener, one sip at a time! \
Our new {product_name}, designed for {target_audience}, not \
only looks good but does good. Made entirely from recycled \
materials, it keeps your drink icy cold for a whole day. \
Make a difference with every drink. Order yours today and \
take a step towards a sustainable future!"

Role of the model: You are a creative director who specializes \
in eco-friendly products and has extensive experience crafting \
compelling marketing messages for a social media-savvy audience.
"""

# Call the OpenAI API to generate the script
response = openai.Completion.create(
    engine="text-davinci-003",
    prompt=prompt,
    max_tokens=300,
    temperature=0.7
)
```

NOVA IMS Information Management School

7.4

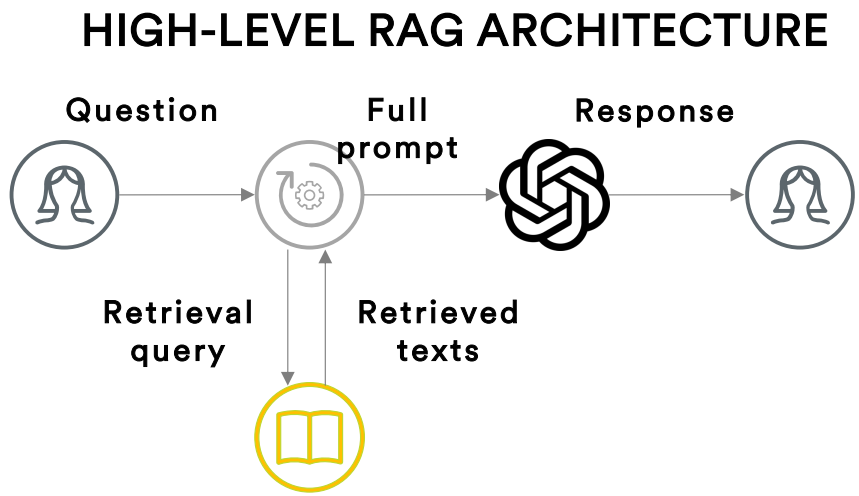
Retrieval-augmented Generation (RAG)

When models access external knowledge sources

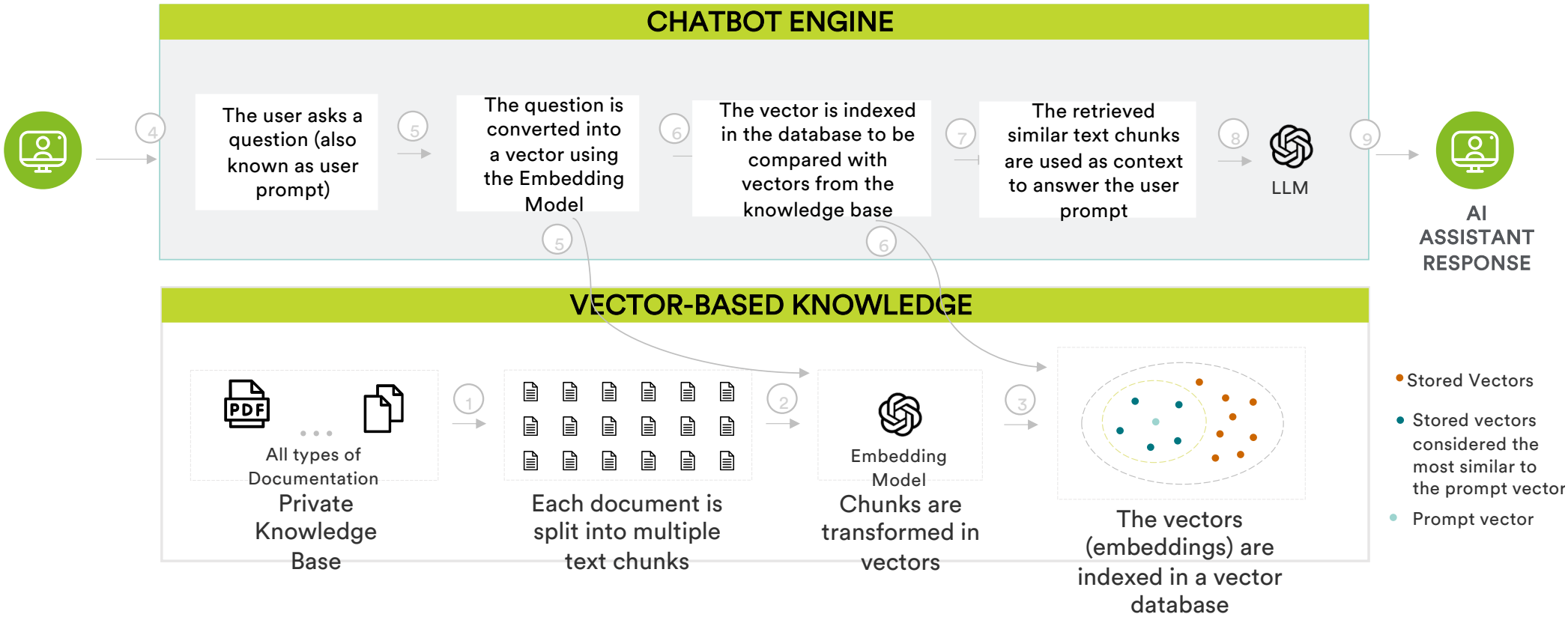
Retrieval-augmented Generation (RAG)

RAG is a technique that combines the generative capabilities of LLMs with the retrieval of external information to produce richer and more accurate responses.

It enhances LLMs' context-awareness by dynamically **integrating relevant data** during the generation process.



Retrieval-augmented Generation (RAG)



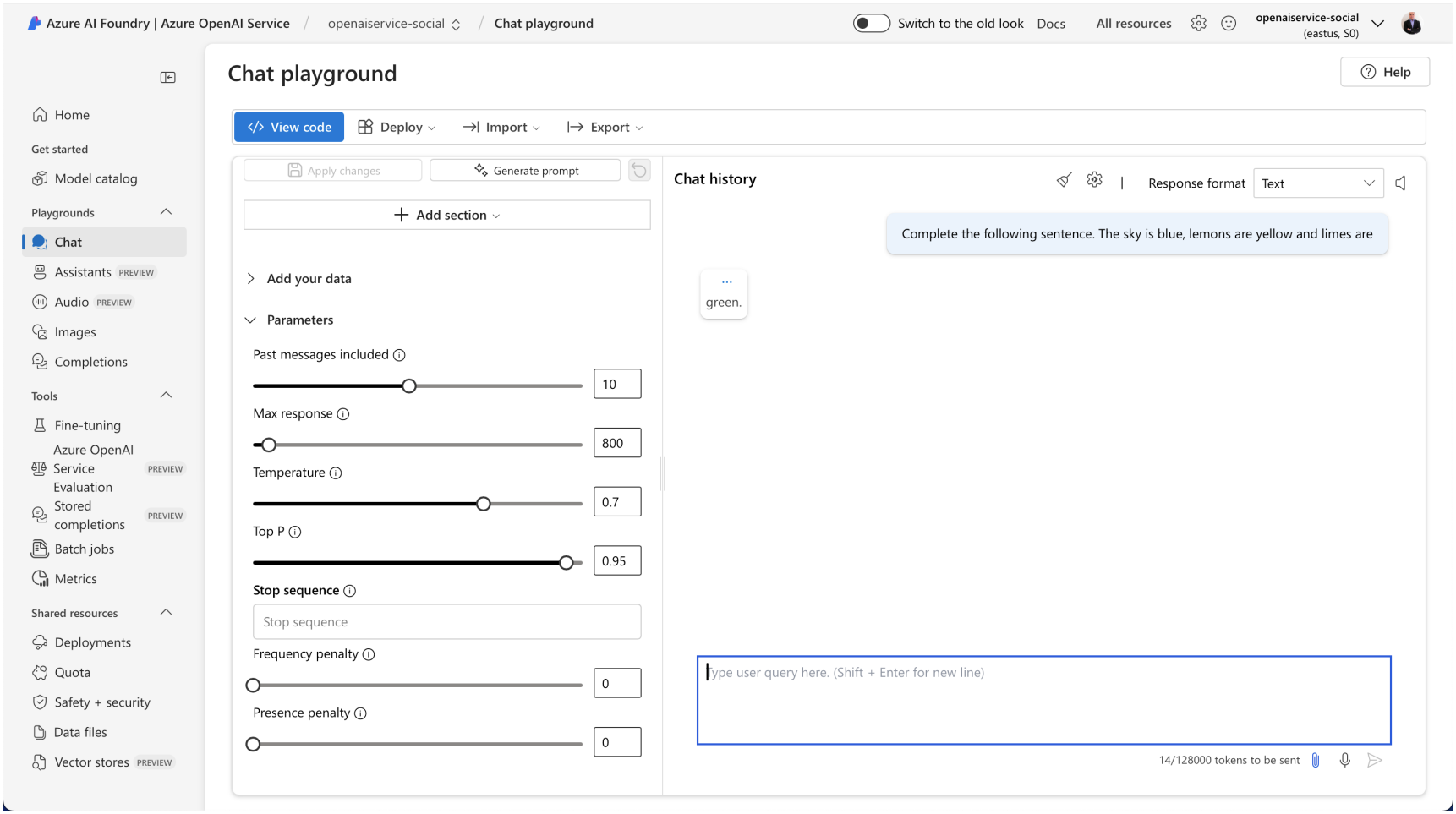
NOVA IMS Information Management School

7.5

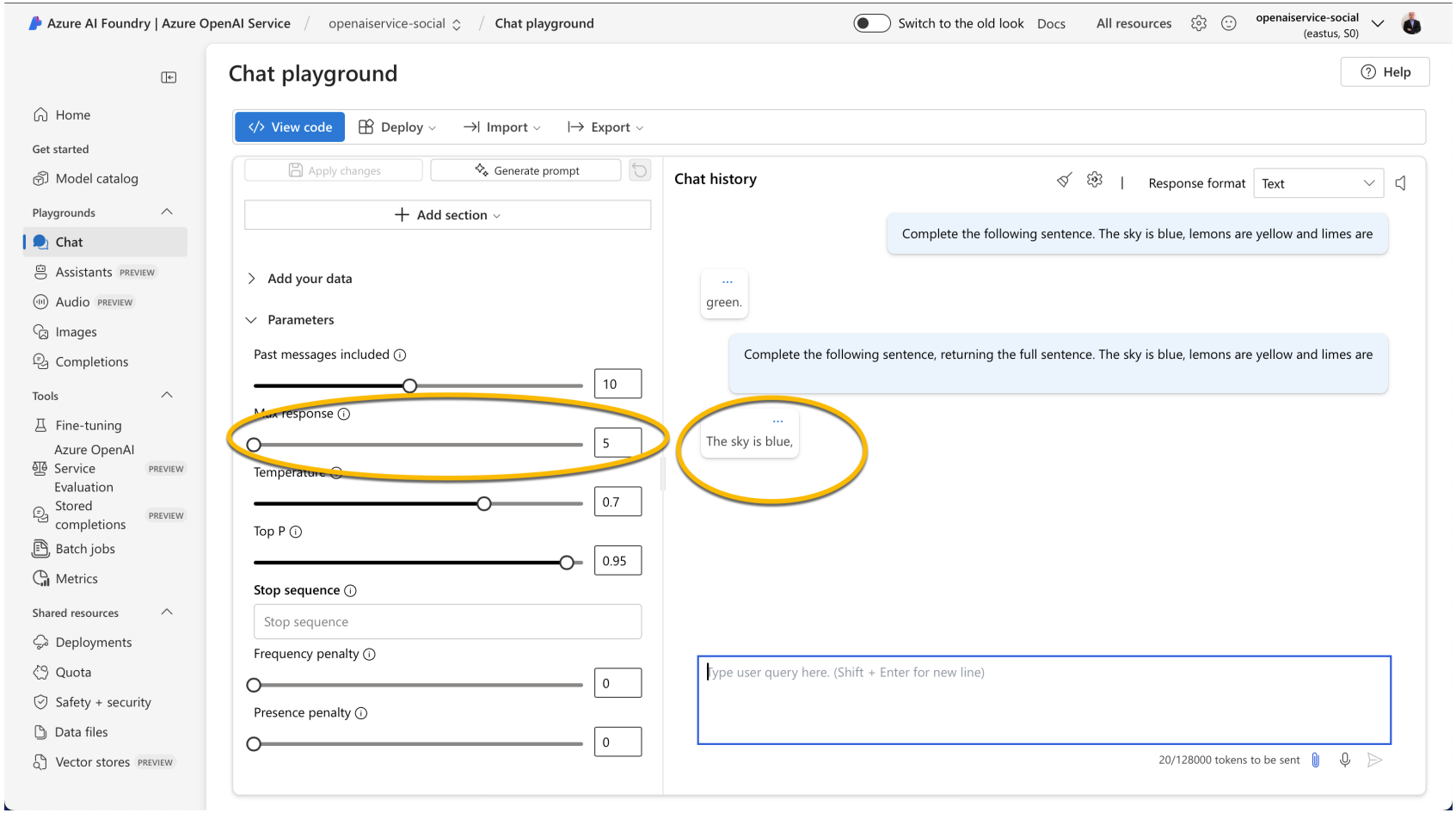
Parameters tuning

Tuning models' outputs through parameter tuning

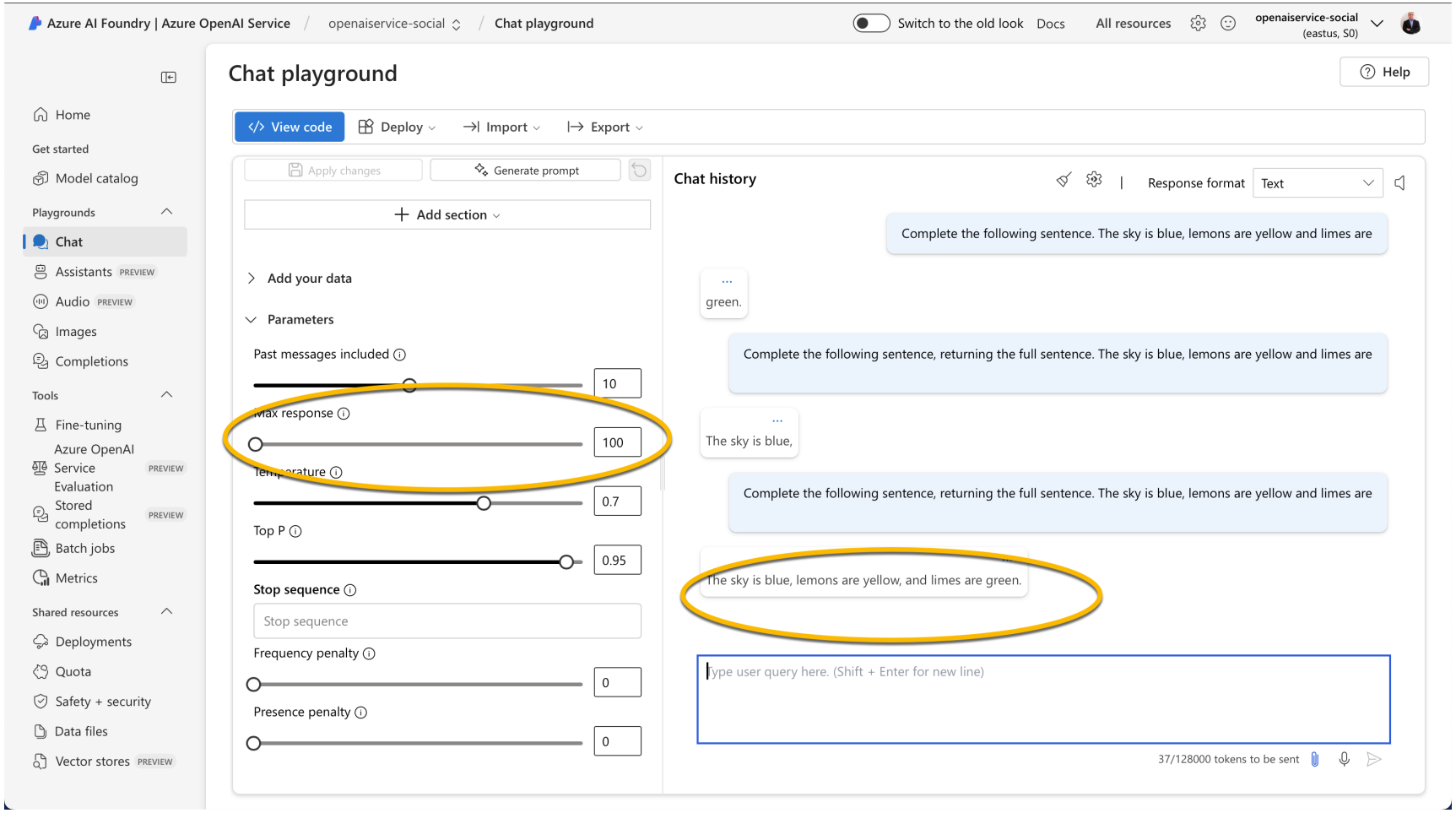
Simple prompts, return simple answers



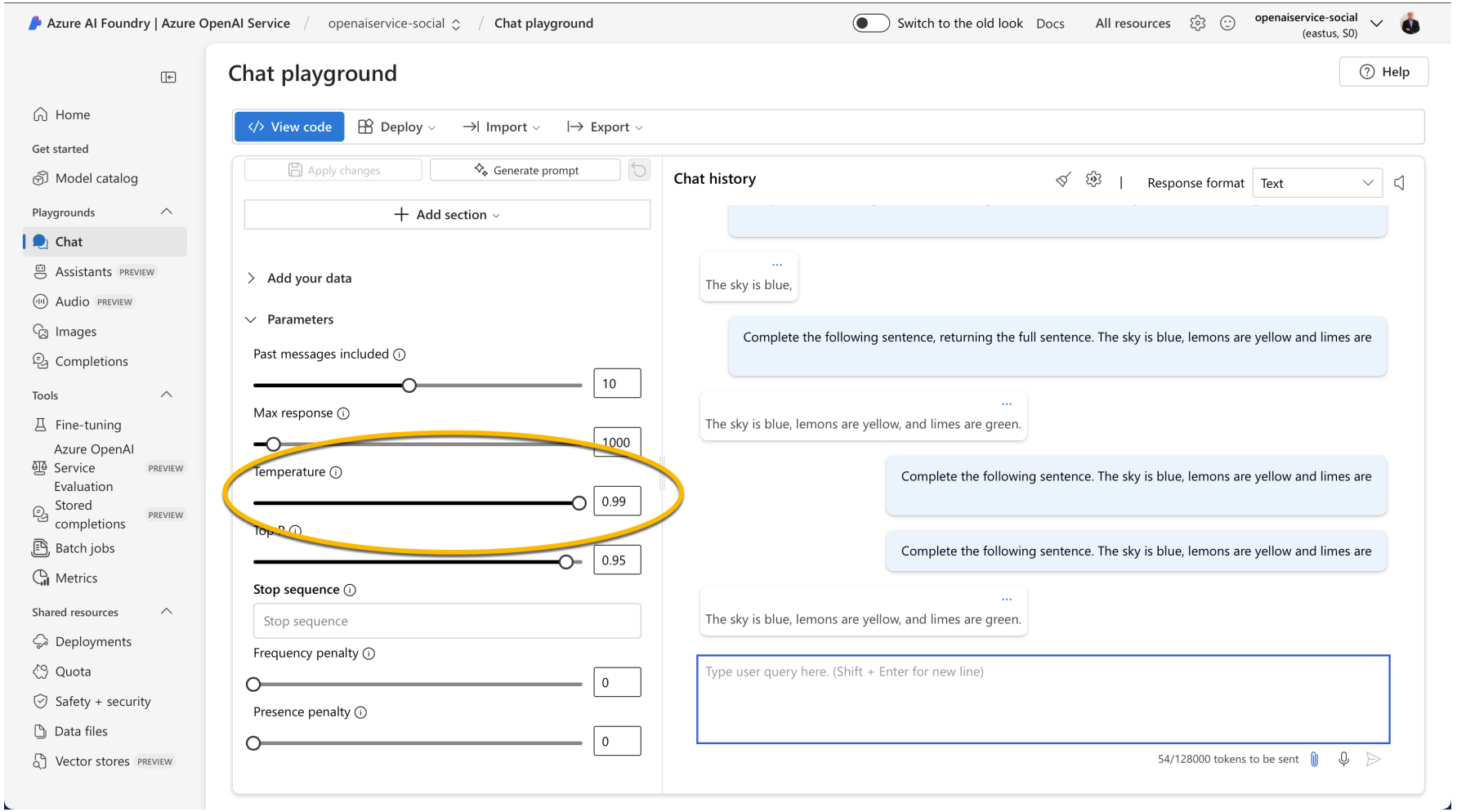
Let the model know when to stop



Let the model know when to stop



Temperature controls creativity



Other parameters

- **Top P**: similar to temperature. Controls randomness but uses a different method. Lowering Top P will narrow the model's token selection to likelier tokens. Increasing Top P will let the model choose from tokens with both high and low likelihood. Try adjusting temperature or Top P but not both
- **Stop sequence**: Make the model end its response at a desired point. The model response will end before the specified sequence, so it won't contain the stop sequence text
- **Frequency penalty**: Reduce the chance of repeating a token proportionally based on how often it has appeared in the text so far. This decreases the likelihood of repeating the exact same text in a response
- **Presence penalty**: Reduce the chance of repeating any token that has appeared in the text at all so far. This increases the likelihood of introducing new topics in a response (good for creativity)

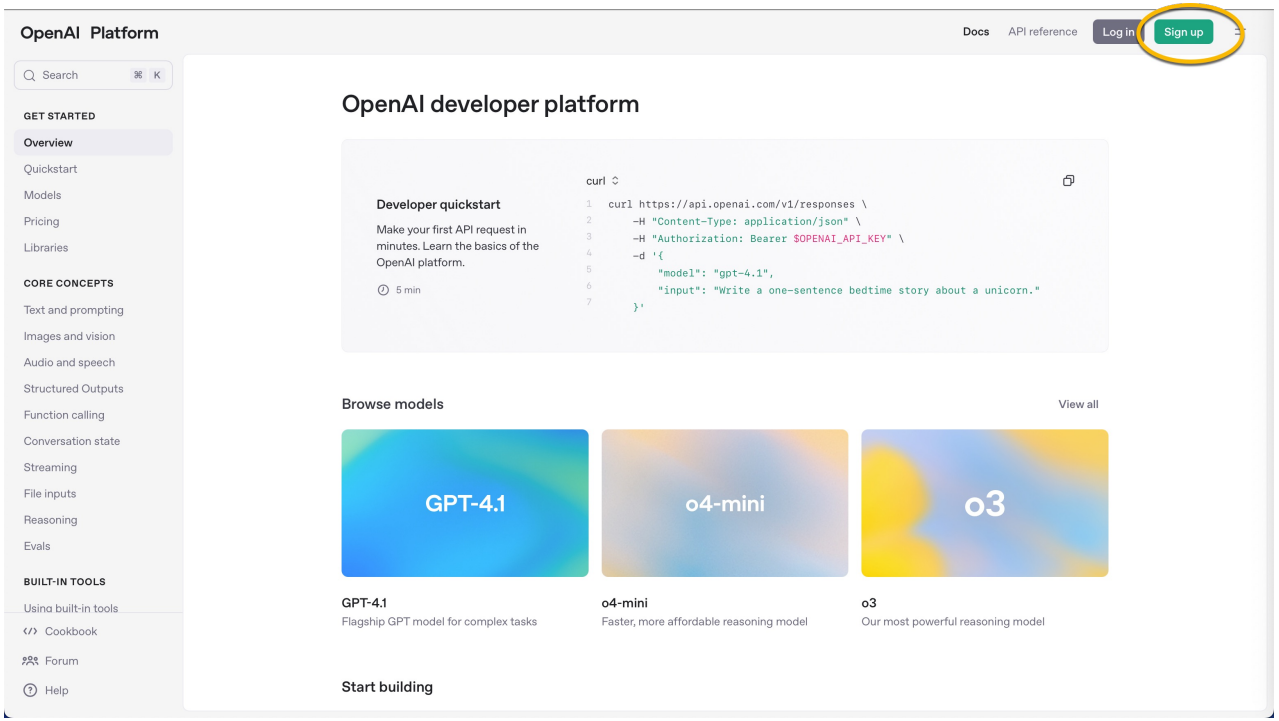
NOVA IMS Information Management School

7.6

ChatGPT: Getting an API key

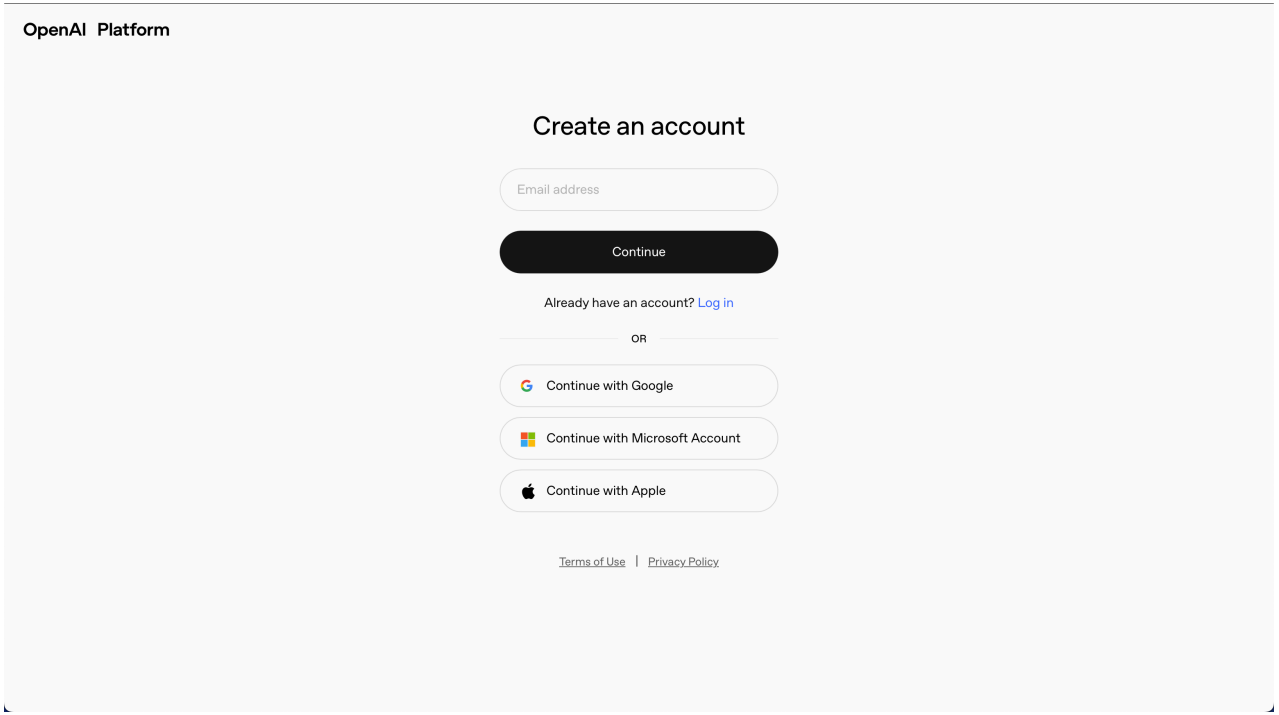
Creating an account in Open AI's platform

- Open a browser and navigate to https://platform.openai.com
- Click **Sign up**



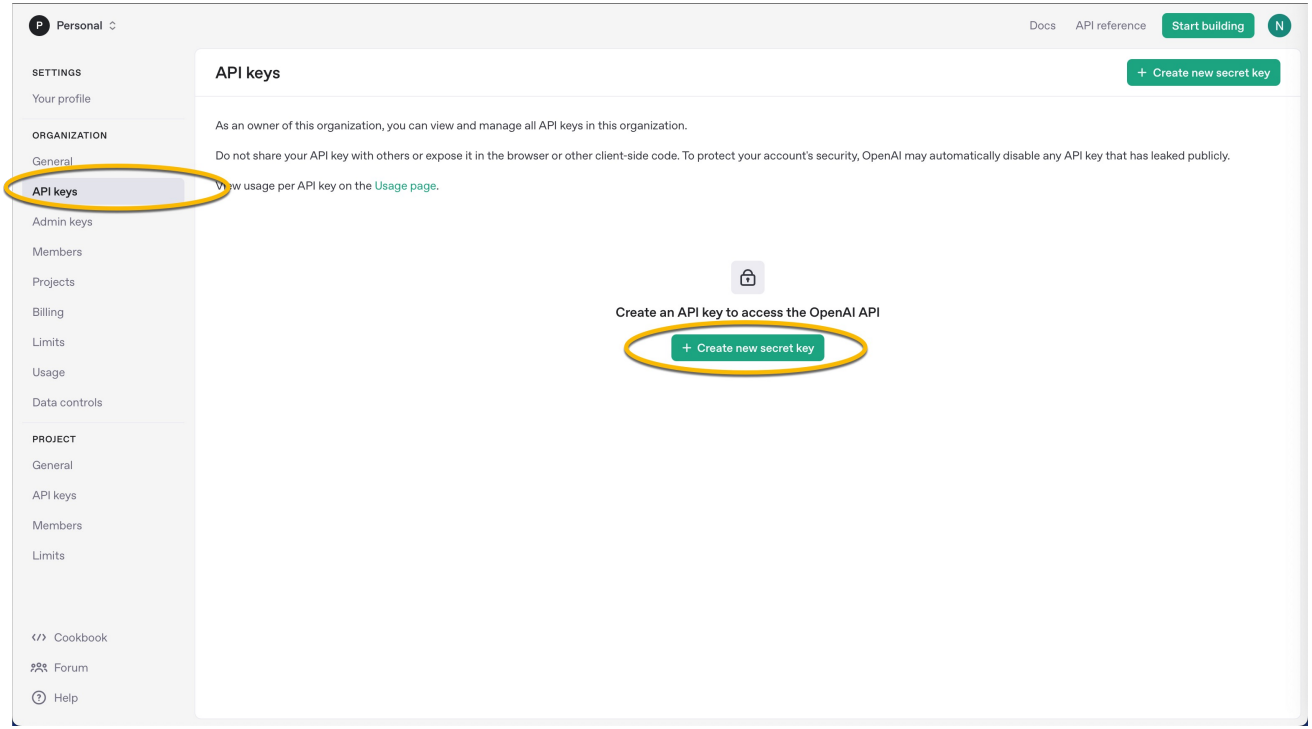
Creating an account in Open AI's platform

- Enter your email address or select an Identity Manager
- Click **Continue** and follow the steps



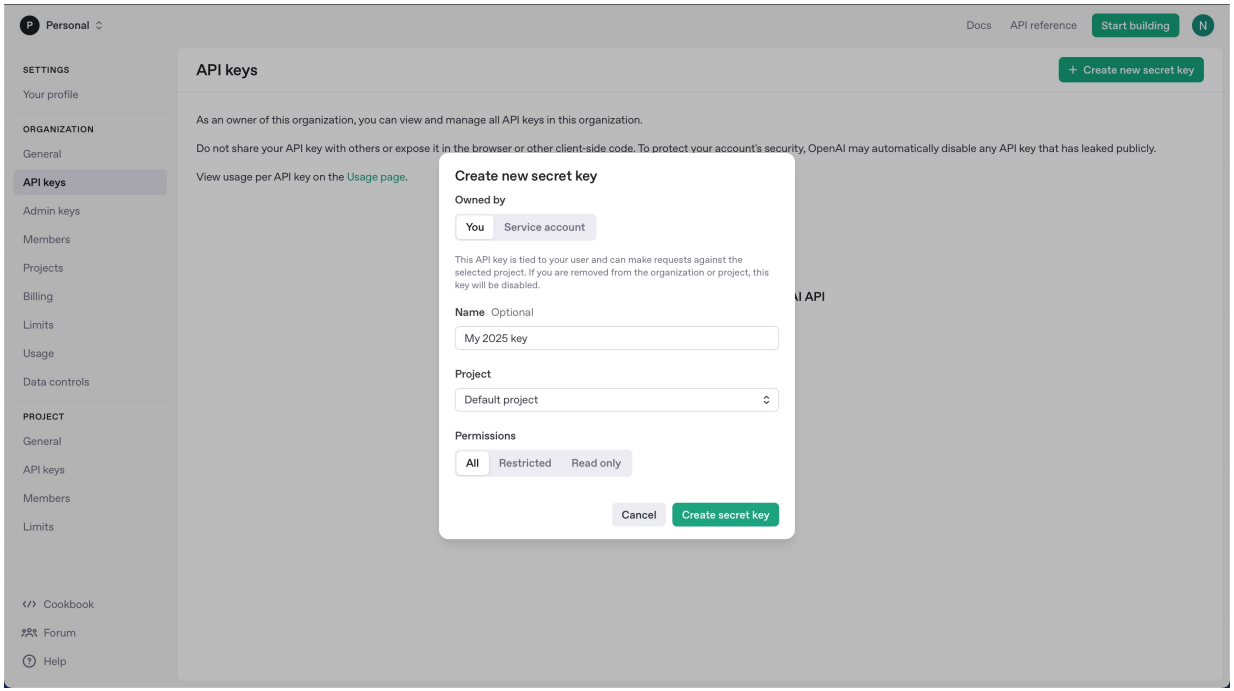
Manage your organization settings and API keys

- Navigate to https://platform.openai.com/settings/organization/api-keys
- Click **Create new secret key** in **API keys**



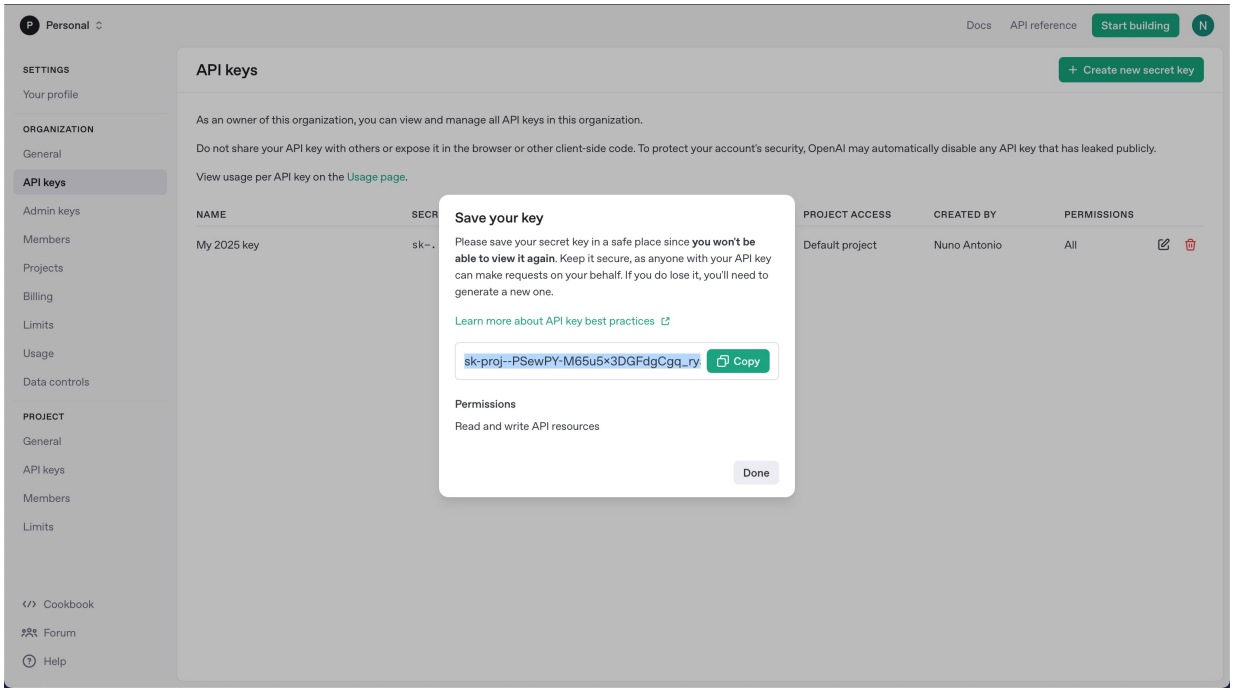
Creating an account in Open AI's platform

- Enter a name for your key (e.g., My Key)
- Click **Create new secret key**



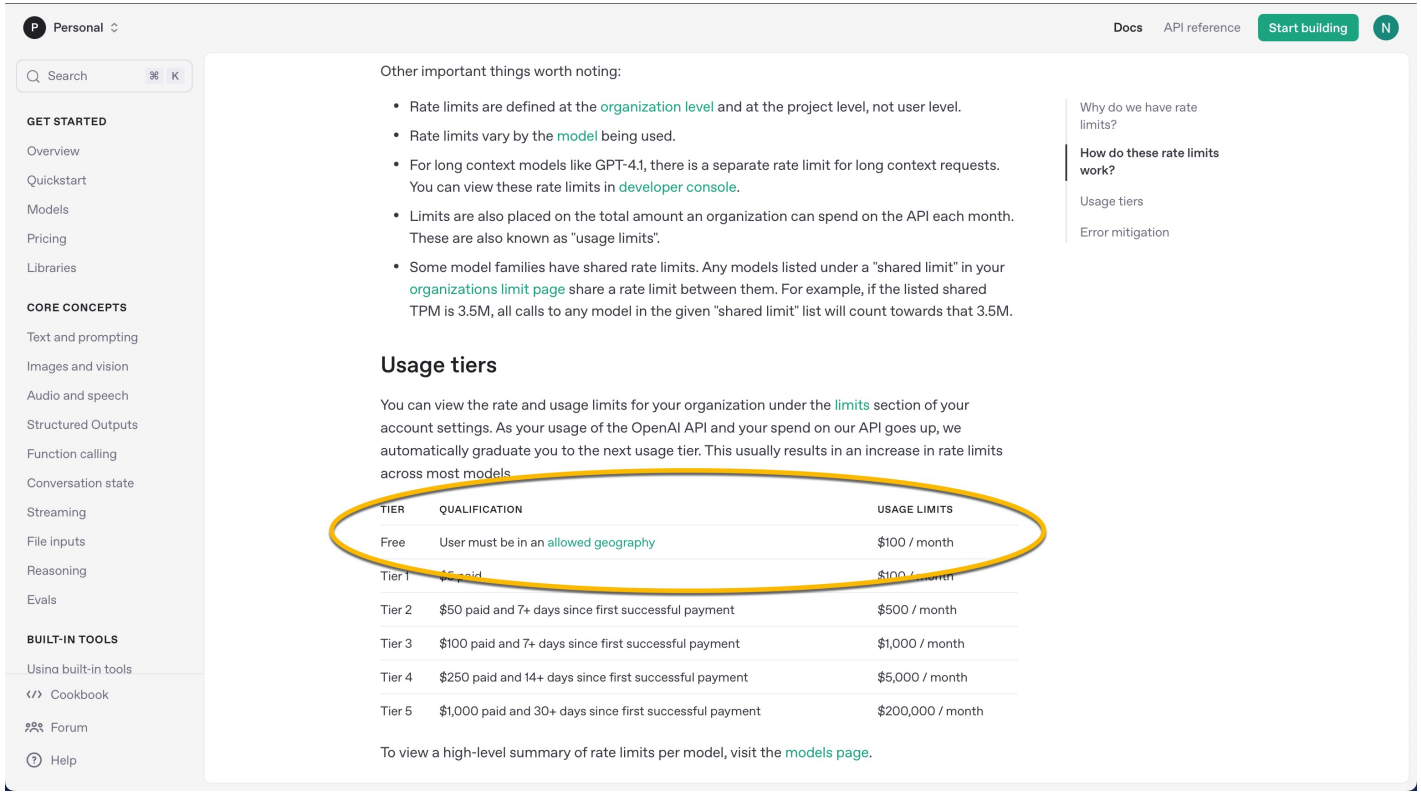
Creating an account in Open AI's platform

- Copy the new key to memory and store the key in a safe place
- Click **Done** to finish



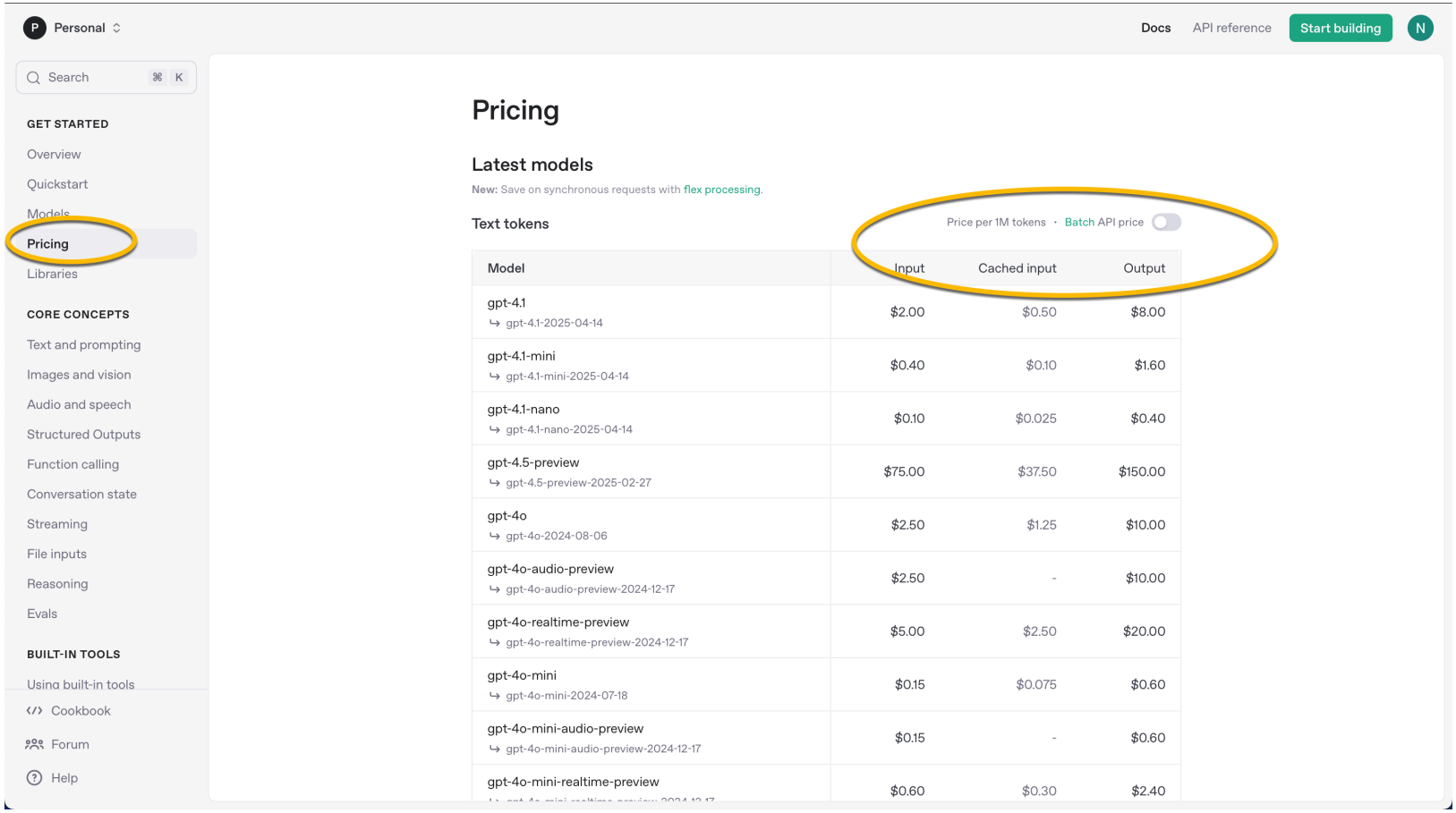
Creating an account in Open AI's platform

- Navigate to https://platform.openai.com/docs/guides/rate-limits/usage-tiers?context=tier-free to check for the Free tier rate limits



Pricing

- The cost differs per model



Open AI current models (May 2025)

Model	Context window	Description
GPT-4.1	1 047 576 tokens	Flagship multimodal model; unmatched long-document handling, strongest reasoning, robust function/tool calling, excels on text + images.
GPT-4o (Omni)	128 000 tokens	Multimodal, fastest Tier-1 model; top performance in non-English, vision, and code tasks, beating GPT-4 Turbo on speed and accuracy.
o4-mini	200 000 tokens	Reasoning-optimized; superior on math/science, deep logical chains, parallel tool execution, lower cost than GPT-4.1 for complex work.
GPT-4 Turbo (2024-04-09)	128 000 tokens	Production workhorse; balanced price/speed, vision support, JSON mode, reliable for chat, code, and general tasks.
GPT-4o-mini	128 000 tokens	Lightweight Omni variant; multimodal, very fast and inexpensive—ideal drop-in upgrade for GPT-3.5-scale workloads.

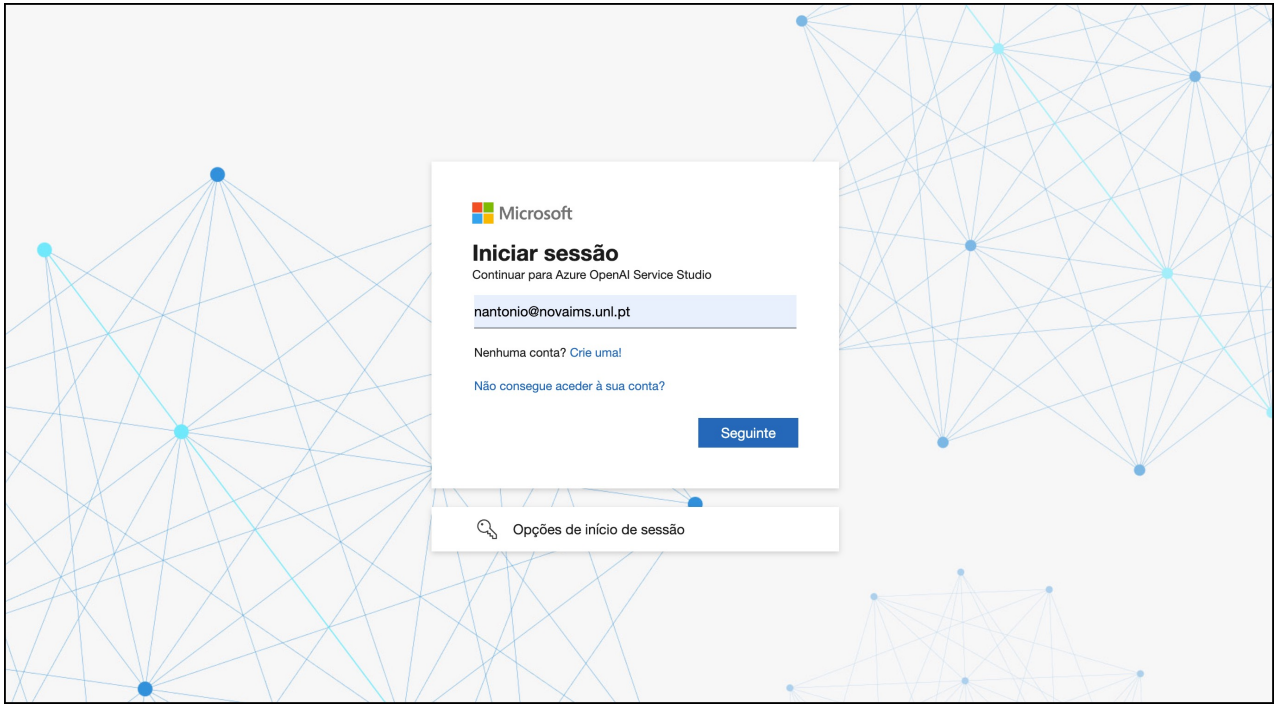
NOVA IMS Information Management School

7.7

Azure Open AI

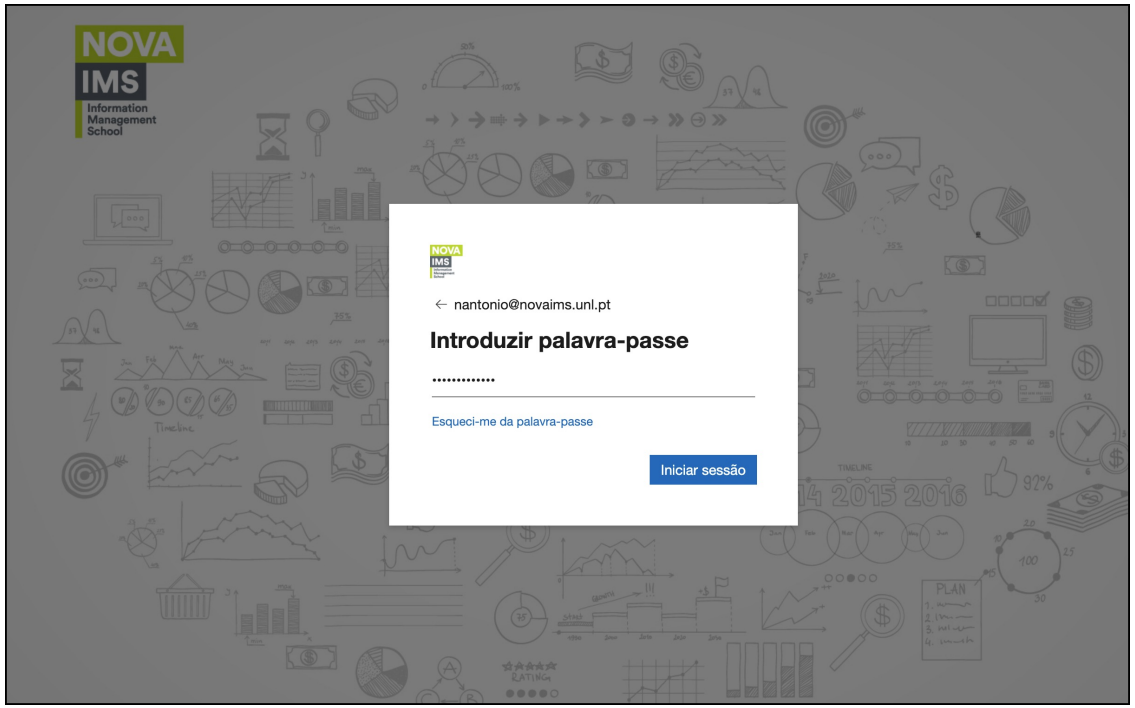
Using NOVA IMS' Azure Open AI service

- Open a browser and navigate to https://oai.azure.com/portal
- Enter you NOVA IMS email address and click Next



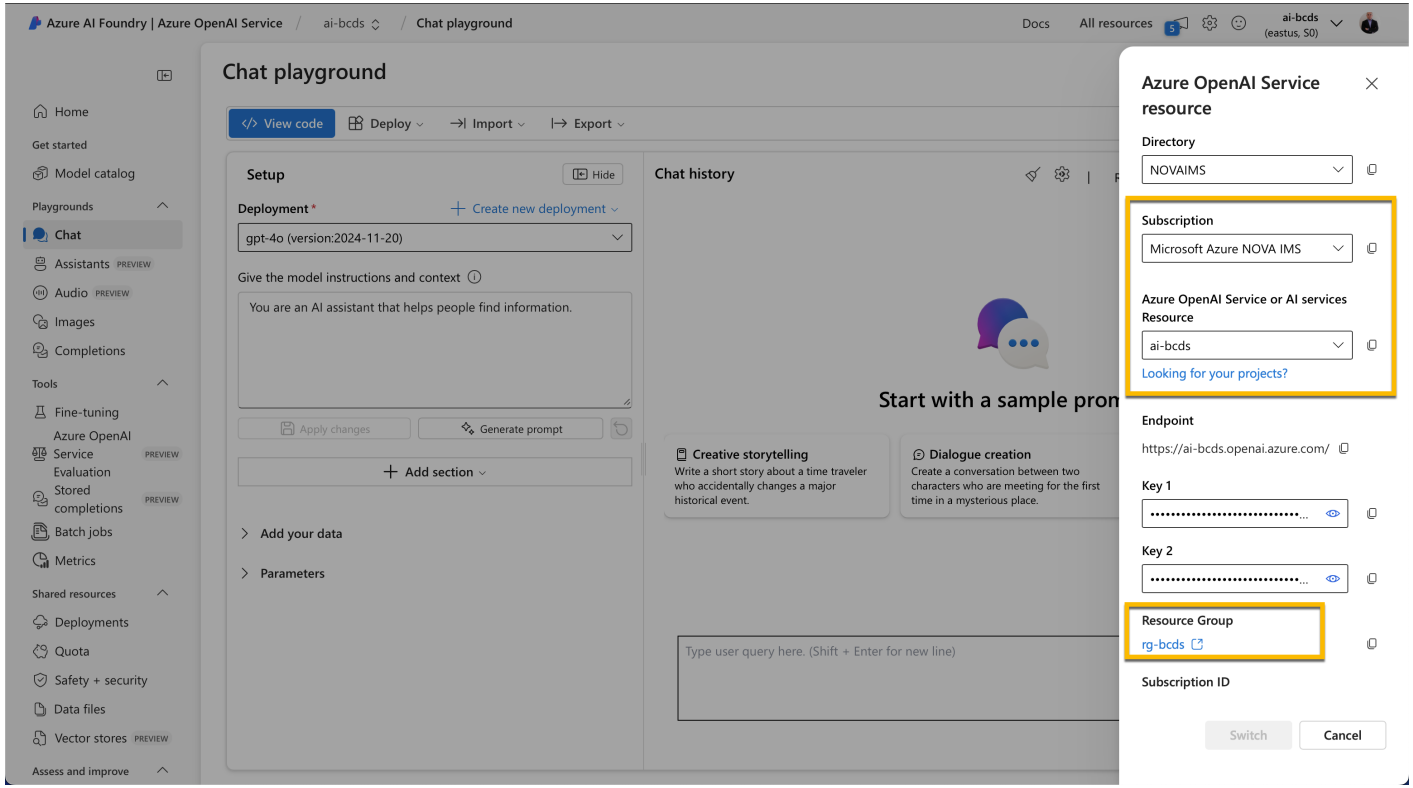
Using NOVA IMS' Azure Open AI service

- Enter your NOVA IMS password and click Next
- In the following window, click to maintain the session opened



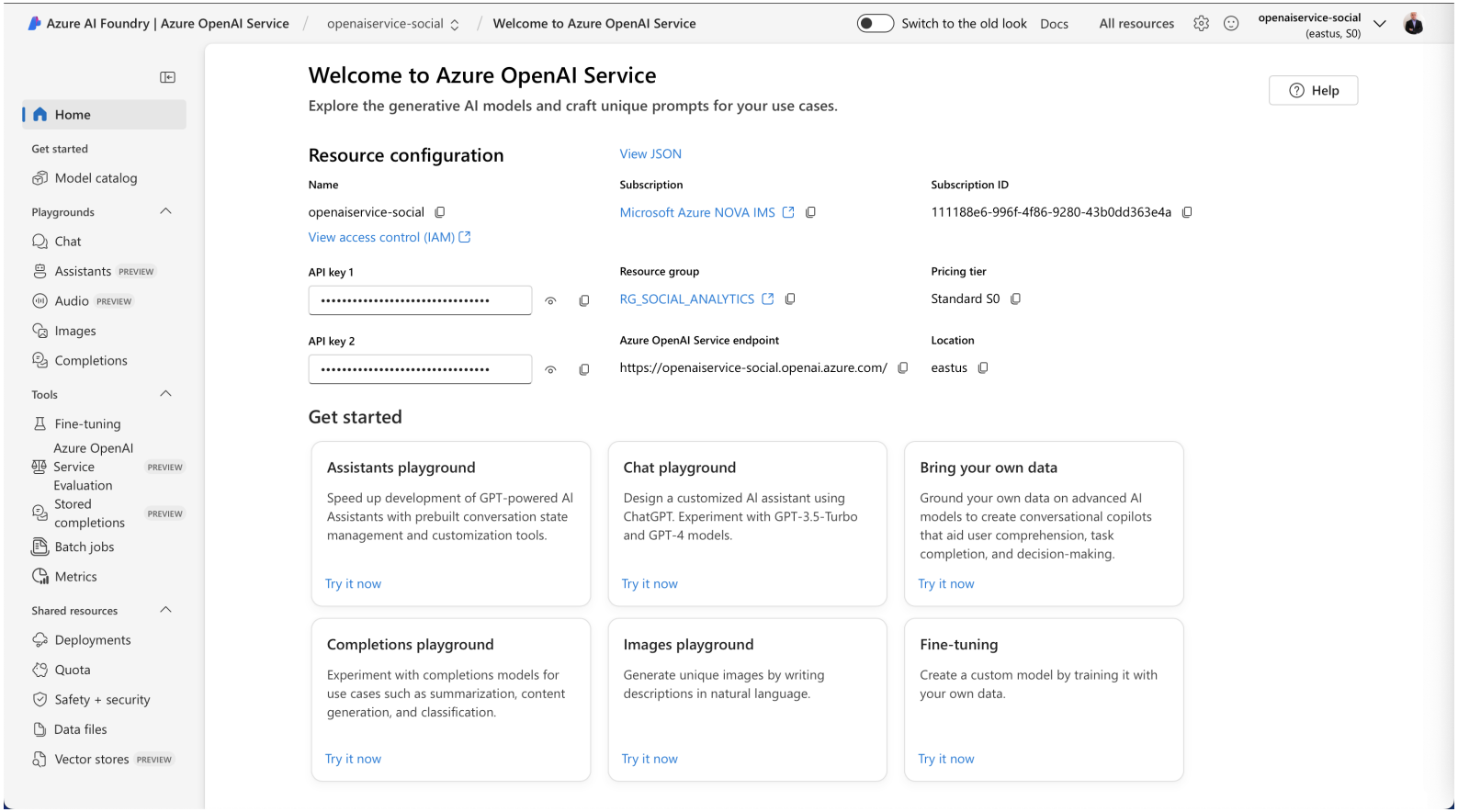
Select the resource for classes

- Select the **Subscription** “Microsoft Azure NOVA IMS” and **Resource** “ai-bcds” (the **Resource Group** is “rg-bcds”)

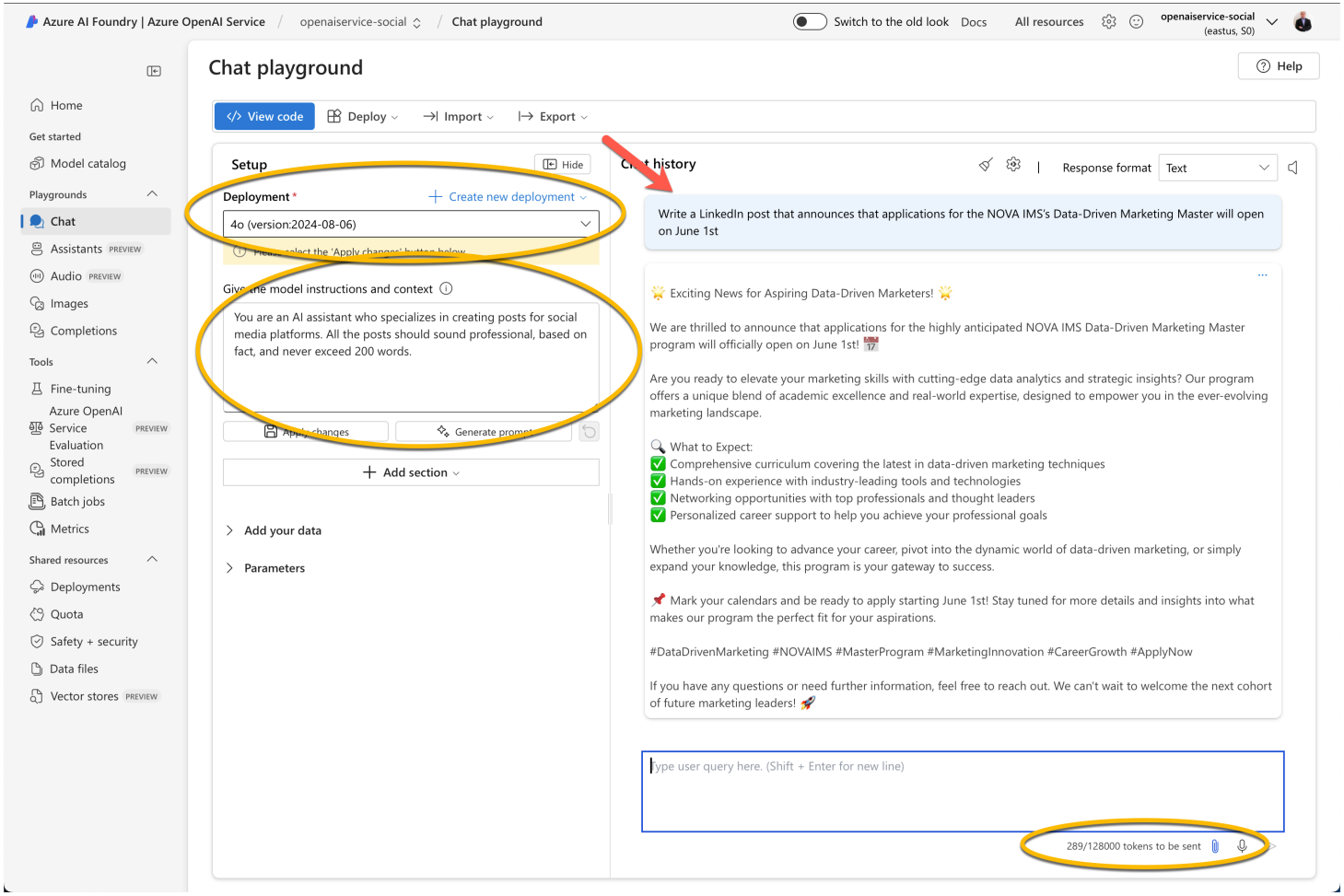


Azure Open AI

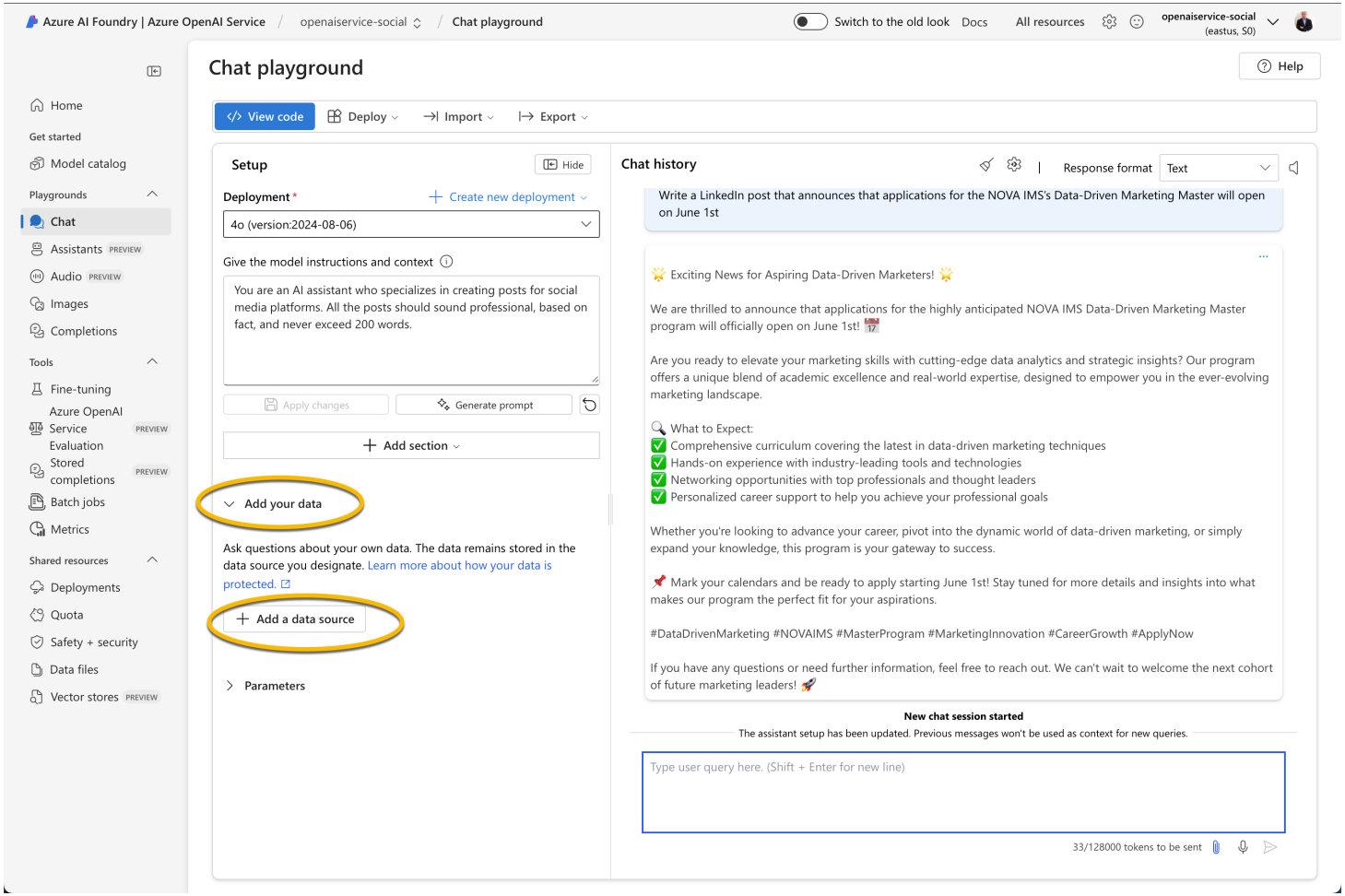
Explore the functionalities



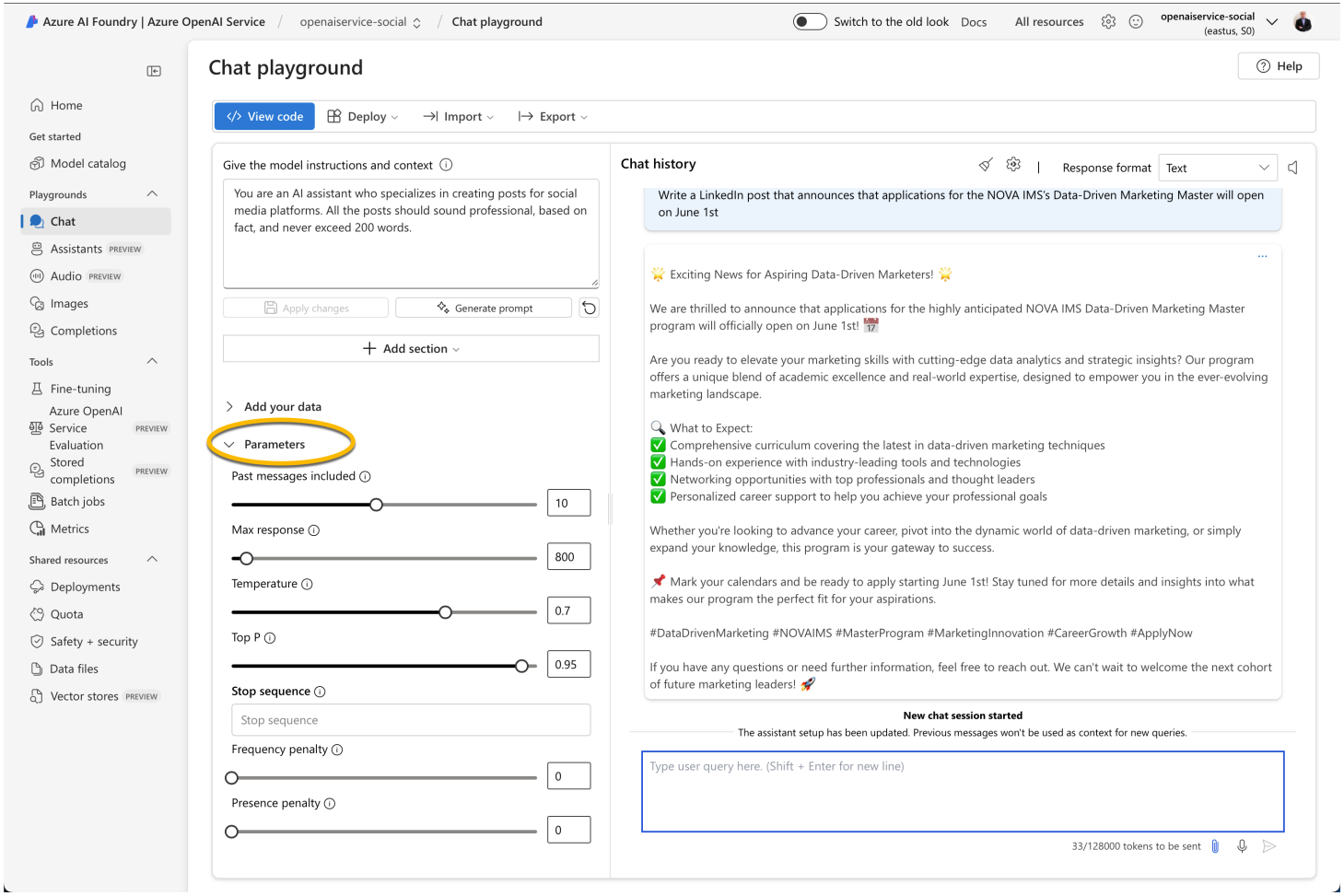
Use the Chat Playground to use GPT 4o



You can add your own data

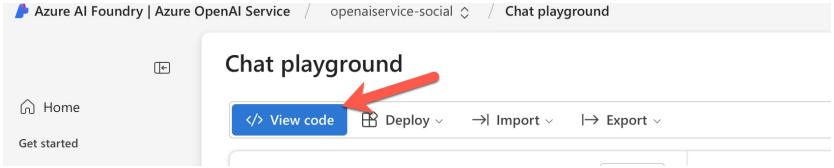


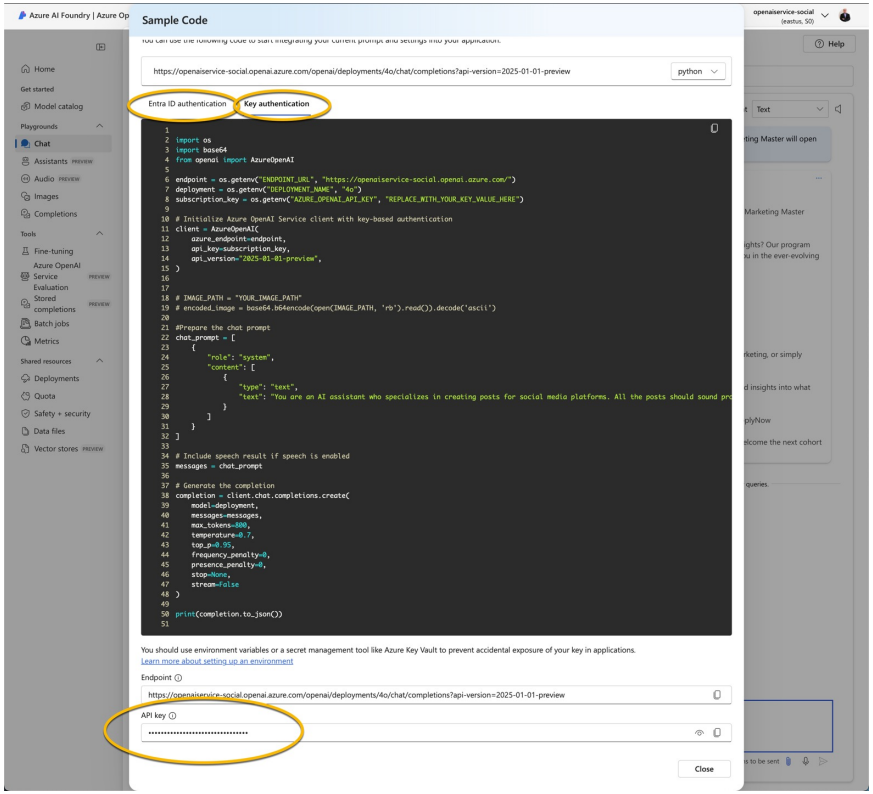
Tune parameters



Obtaining the Python code

- Click the **View code** to check the Python code to call
- Get your API token in the link bellow the code
- You can copy the code by clicking the **Copy** button





NOVA IMS Information Management School

References

- https://github.com/Mooler0410/LLMsPracticalGuide
- https://www.slideshare.net/rmcdermo/fhllmtalk20231005pdf
- https://learn.microsoft.com/en-us/training/modules/introduction-large-language-models/2-understand-large-language-models
- https://developer.nvidia.com/blog/how-to-get-better-outputs-from-your-large-language-model/
- https://en.wikipedia.org/wiki/Prompt_engineering
- https://learn.microsoft.com/en-us/azure/ai-services/openai/reference

Business Cases with Data Science

Acreditações e Certificações

Instituto Superior de Estatística e Gestão da Informação
Universidade Nova de Lisboa