EXPERT INSIGHT

Early Access

Practical Generative AI with ChatGPT

Start your generative AI journey by using ChatGPT to maximize your productivity and creativity

Second Edition

Valentina Alto

packt

Practical Generative AI with ChatGPT

Copyright © 2024 Packt Publishing

All rights reserved. No part of this book may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, without the prior written permission of the publisher, except in the case of brief quotations embedded in critical articles or reviews.

Every effort has been made in the preparation of this book to ensure the accuracy of the information presented. However, the information contained in this book is sold without warranty, either express or implied. Neither the author, nor Packt Publishing, and its dealers and distributors will be held liable for any damages caused or alleged to be caused directly or indirectly by this book.

Packt Publishing has endeavored to provide trademark information about all of the companies and products mentioned in this book by the appropriate use of capitals. However, Packt Publishing cannot guarantee the accuracy of this information.

Early Access Publication: Practical Generative AI with ChatGPT

Early Access Production Reference: B31559

Published by Packt Publishing Ltd.

Grosvenor House

11 St Paul's Square

Birmingham

B3 1RB, UK

ISBN: 978-1-83664-785-0

www.packt.com

Table of Contents

Practical Generative AI with ChatGPT, Second Edition: Start your generative AI journey by using ChatGPT to maximize your productivity and creativity

1. Introduction to Generative AI
    - I. Join our book community on Discord
    - II. Introducing generative AI
        - i. Domains of generative AI
        - ii. Text generation
        - iii. Image generation
        - iv. Music generation
        - v. Video generation
    - III. The history and current status of research
    - IV. 18 months later: main trends and innovations
        - i. Retrieval Augmented Generation
        - ii. Multimodality
        - iii. AI Agents
        - iv. Small Language Models
    - V. Summary
    - VI. References
2. OpenAI and ChatGPT: Beyond the Market Hype
    - I. Join our book community on Discord
    - II. Technical requirements
    - III. What is OpenAI?
    - IV. An overview of OpenAI Playground
        - i. OpenAI Models Families
        - ii. Trying OpenAI models in the Playground
    - V. ChatGPT: the state of the art
    - VI. Summary

VII. References
3. Understanding Prompt Design
I. Join our book community on Discord
II. What is prompt engineering?
III. Zero-, one-, and few-shot learning – typical of transformers models
IV. Principles of prompt engineering
i. Clear Instructions
ii. Split complex tasks into subtasks
iii. Ask for justification
iv. Generate many outputs, then use the model to pick the best one
v. Use delimiters
V. Advanced techniques
i. Chain of Thoughts
ii. ReAct
VI. Avoiding the risk of hidden bias and taking into account ethical considerations in ChatGPT
VII. Summary
VIII. References
4. Boosting Day-to-Day Productivity with ChatGPT
I. Join our book community on Discord
II. Technical requirements
III. ChatGPT as a daily assistant
IV. Generating text
V. Improving writing skills and translation
VI. Quick information retrieval and competitive intelligence
VII. Summary
5. Developing the Future with ChatGPT
I. Join our book community on Discord
II. Why ChatGPT for developers?
III. Generating, optimizing, and debugging code

IV. Generating documentation and code explainability
V. Understanding ML model interpretability
VI. Translation among different programming languages
VII. Summary
6. Mastering Marketing with ChatGPT
I. Join our book community on Discord
II. Technical requirements
III. Marketers' need for ChatGPT
IV. New product development and the go-to-market strategy
V. A/B testing for marketing comparison
VI. Boosting Search Engine Optimization (SEO)
VII. Sentiment analysis to improve quality and increase customer satisfaction
VIII. Summary
7. Research Reinvented with ChatGPT
I. Join our book community on Discord
II. Researchers' need for ChatGPT
III. Brainstorming literature for your study
IV. Providing support for the design and framework of your experiment
V. Generating and formatting a bibliography
VI. Generating a presentation of the study
VII. Summary
VIII. References
8. Exploring GPTs
I. Join our book community on Discord
II. Technical requirements
III. What are GPTs?
IV. Personal Assistant
V. Code Assistant
VI. Marketing Assistant

VII. Research Assistant
VIII. Summary
IX. References

Practical Generative AI with ChatGPT, Second Edition: Start your generative AI journey by using ChatGPT to maximize your productivity and creativity

Welcome to Packt Early Access. We're giving you an exclusive preview of this book before it goes on sale. It can take many months to write a book, but our authors have cutting-edge information to share with you today. Early Access gives you an insight into the latest developments by making chapter drafts available. The chapters may be a little rough around the edges right now, but our authors will update them over time.

You can dip in and out of this book or follow along from start to finish; Early Access is designed to be flexible. We hope you enjoy getting to know more about the process of writing a Packt book.

1. Chapter 1: Introduction to Generative AI
2. Chapter 2: Introducing OpenAI and ChatGPT
3. Chapter 3: From Prompt Design to Prompt Engineering
4. Chapter 4: Boosting Day-to-Day Productivity with ChatGPT
5. Chapter 5: Developing the Future with ChatGPT
6. Chapter 6: Mastering Marketing with ChatGPT
7. Chapter 7: Research Reinvented with ChatGPT
8. Chapter 8: Unleashing Creativity Visually with ChatGPT
9. Chapter 9: Exploring GPTs

10. Chapter 10: Leveraging OpenAI’s Models for Enterprise-Scale Applications with Models’ APIs
11. Chapter 11: Trending Use Cases
12. Chapter 12: Epilogue and Final Thoughts

1 Introduction to Generative AI

Join our book community on Discord

https://packt.link/EarlyAccess

Hello! Welcome to *The Ultimate Guide to ChatGPT and OpenAI!* In this book, we will explore the fascinating world of **Generative Artificial Intelligence** (**GAI**) and its groundbreaking applications. Generative AI has transformed the way we interact with machines, enabling computers to create, predict, and learn without explicit human instruction. Since the launch of ChatGPT in November 2023, we have witnessed unprecedented advances in natural language processing, image and video synthesis, and many other fields. Whether you are a curious beginner or an experienced practitioner, this guide will equip you with the knowledge and skills to navigate the exciting landscape of generative AI. So, let's dive in and start with some definitions of the context we are moving in.

This chapter provides an overview of the field of generative AI, which consists of creating new and unique data or content using **machine learning** (**ML**) algorithms.

It focuses on the applications of generative AI to various fields, such as image synthesis, text generation, and music composition, highlighting the potential of generative AI to revolutionize various industries. This introduction to generative AI will provide context for where this technology lives, as well as the knowledge to collocate it within the wide world of AI, ML, and **Deep Learning** (**DL**). Then, we will dwell on the main areas of applications of generative AI with concrete examples and recent developments so that you can get familiar with the impact it may have on businesses and society in general.

Also, being aware of the research journey toward the current state of the art of generative AI will give you a better understanding of the foundations of recent developments and state-of-the-art models.

All this, we will cover with the following topics:

- Understanding generative AI
- Exploring the domains of generative AI
- The history and current status of research on generative AI

By the end of this chapter, you will be familiar with the exciting world of generative AI, its applications, the research history behind it, and the current developments, which could have – and are currently having – a disruptive impact on businesses.

Introducing generative AI

AI has been making significant strides in recent years, and one of the areas that has seen considerable growth is generative AI. Generative AI is a subfield of AI and DL that focuses on generating new content, such as images, text, music, and video, by using algorithms and models that have been trained on existing data using ML techniques.

In order to better understand the relationship between AI, ML, DL, and generative AI, consider AI as the foundation, while ML, DL, and generative AI represent increasingly specialized and focused areas of study and application:

- AI represents the broad field of creating systems that can perform tasks, showing human intelligence and ability and being able to interact with the ecosystem.
- ML is a branch that focuses on creating algorithms and models that enable those systems to learn and improve themselves with time and training. ML models learn from existing data and automatically update their parameters as they *grow*.
- DL is a sub-branch of ML, in the sense that it encompasses deep ML models. Those deep models are called **neural networks** and are particularly suitable in domains such as **computervision** or **Natural Language Processing** (**NLP**). When we talk about ML and DL models, we typically refer to discriminative models, whose aim is that of making predictions or inferencing patterns on top of data.
- And finally, we get to generative AI, a further sub-branch of DL, which doesn’t use deep Artificial Neural Networks to cluster, classify, or make predictions on existing data: it uses those powerful Artificial Neural Network models to generate

brand new content, from images to natural language, from music to video.

The following figure shows how these areas of research are related to each other:

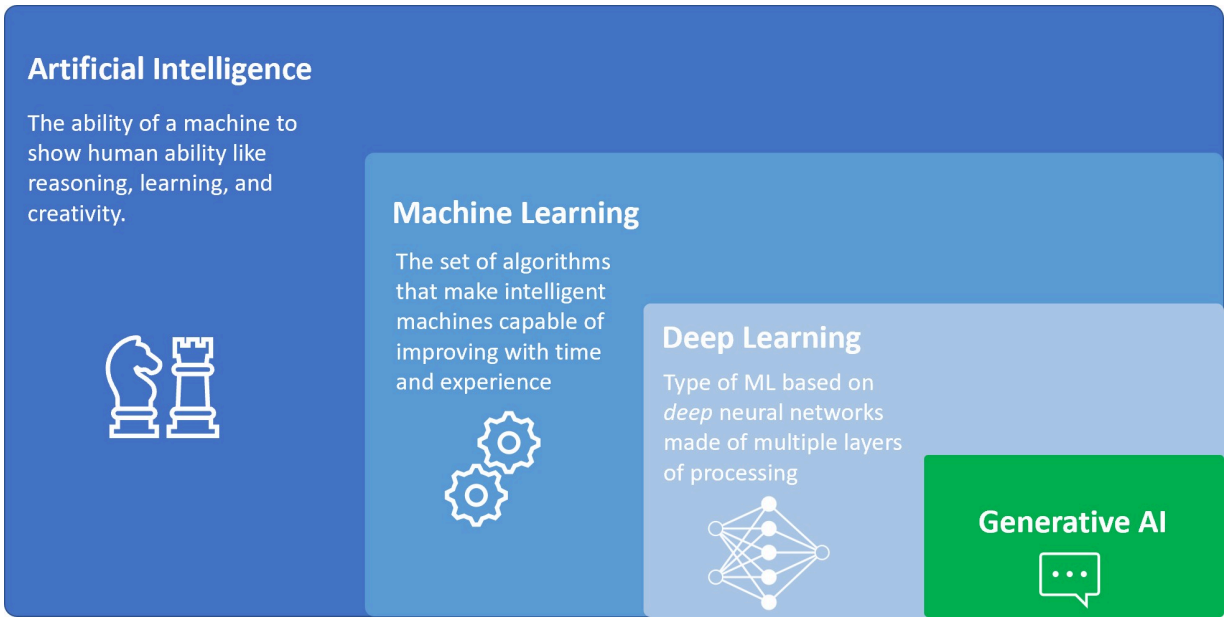


Figure 1.1 – Relationship between AI, ML, DL, and generative AI

Generative AI models can be trained on vast amounts of data and then they can generate new examples from scratch using patterns in that data. This generative process is different from discriminative models, which are trained to predict the class or label of a given example.

The type of Neural Networks that feature Generative AI are called Large Foundation Models LFMs) and, in the case of language models like ChatGPT, we talk about **Large Language Models** (**LLMs**).

Large Language Models are a type of Artificial Neural Network featured by a transformer architecture. They are

characterized by a huge number of parameters (in the order of trillions of parameters) and have been trained on billions of words. Given the training set, LLMs are capable of understanding and generating natural language given user's input.

Even though text understanding and generation is probably one of the most outstanding features of Generative AI, this field covers many domains.

Domains of generative AI

In recent years, generative AI has made significant advancements and has expanded its applications to a wide range of domains, such as art, music, fashion, architecture, and many more. In some of them, it is indeed transforming the way we create, design, and understand the world around us. In others, it is improving and making existing processes and operations more efficient.

The fact that generative AI is used in many domains also implies that its models can deal with different kinds of data, from natural language to audio or images. Let us understand how generative AI models address different types of data and domains.

Text generation

One of the greatest applications of generative AI—and the one we are going to cover the most throughout this book—is its capability to produce new content in natural language. Indeed, generative AI algorithms can be used to generate new text, such as articles, poetry, and product descriptions.

For example, a language model such as GPT-4o, developed by OpenAI, can be trained on large amounts of text data and then used to generate new, coherent, and grammatically correct text in different languages (both in terms of input and output), as well as extracting relevant features from text such as keywords, topics, or full summaries.

Here is an example of working with GPT-3:

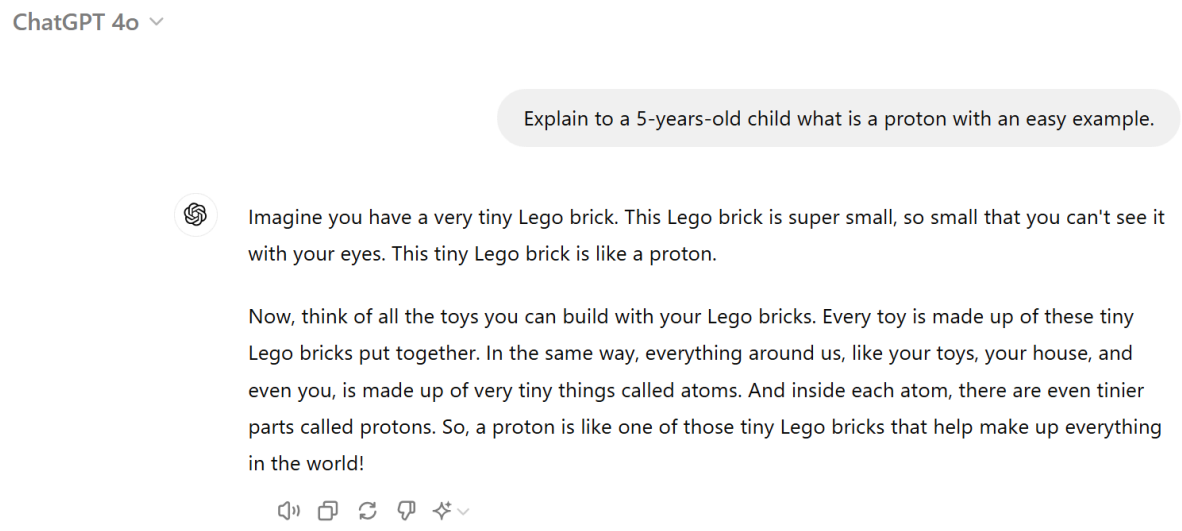


Figure 1.2 – Example of ChatGPT responding to a user prompt, also adding references

Next, we will move on to image generation.

Image generation

One of the earliest and most well-known examples of generative AI in image synthesis is the **Generative Adversarial Network** (**GAN**) architecture introduced in the 2014 paper by I. Goodfellow et al., *Generative Adversarial Networks*. The purpose of GANs is to generate realistic images that are indistinguishable from real

images. This capability had several interesting business applications, such as generating synthetic datasets for training computer vision models, generating realistic product images, and generating realistic images for virtual reality and augmented reality applications.

Here is an example of faces of people who do not exist since they are entirely generated by AI:

Figure 1.3 – Imaginary faces generated by GAN StyleGAN2 at https://this-person-does-not-exist.com/en

Then, in 2021, a new generative AI model was introduced in this field by OpenAI, **DALL-E**. Different from GANs, the DALL-E model is designed to generate images from descriptions in natural language (GANs take a random noise vector as input) and can generate a wide range of images, which may not look realistic but still depict the desired concepts.

DALL-E has great potential in creative industries such as advertising, product design, and fashion, among others, to create unique and creative images.

Since its first release to date (December 2024), DALL-E has improved dramatically, as you can see in the following examples. Let's see below an artistic creation authored by DALL-E at the dawn of its life:

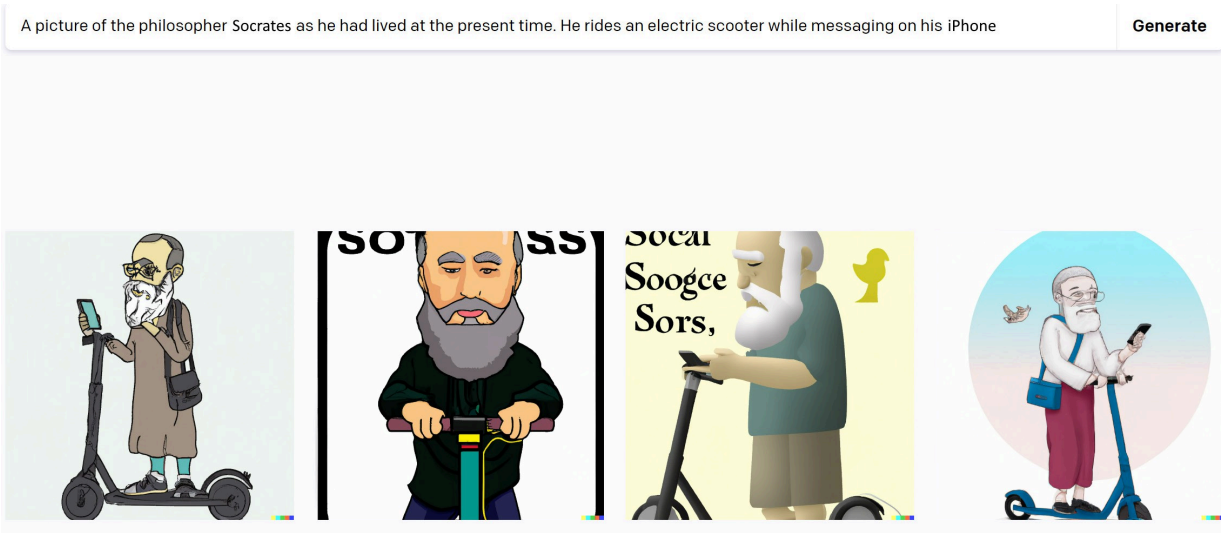


Figure 1.4 – Images generated by DALL-E with a natural language prompt as input

Let's now see what DALL-E3, the most recent version of the model at the time of writing this book, can produce (here I'm using Microsoft Image Creator powered by DALL-E3. You can try it at https://copilot.microsoft.com/images/create):

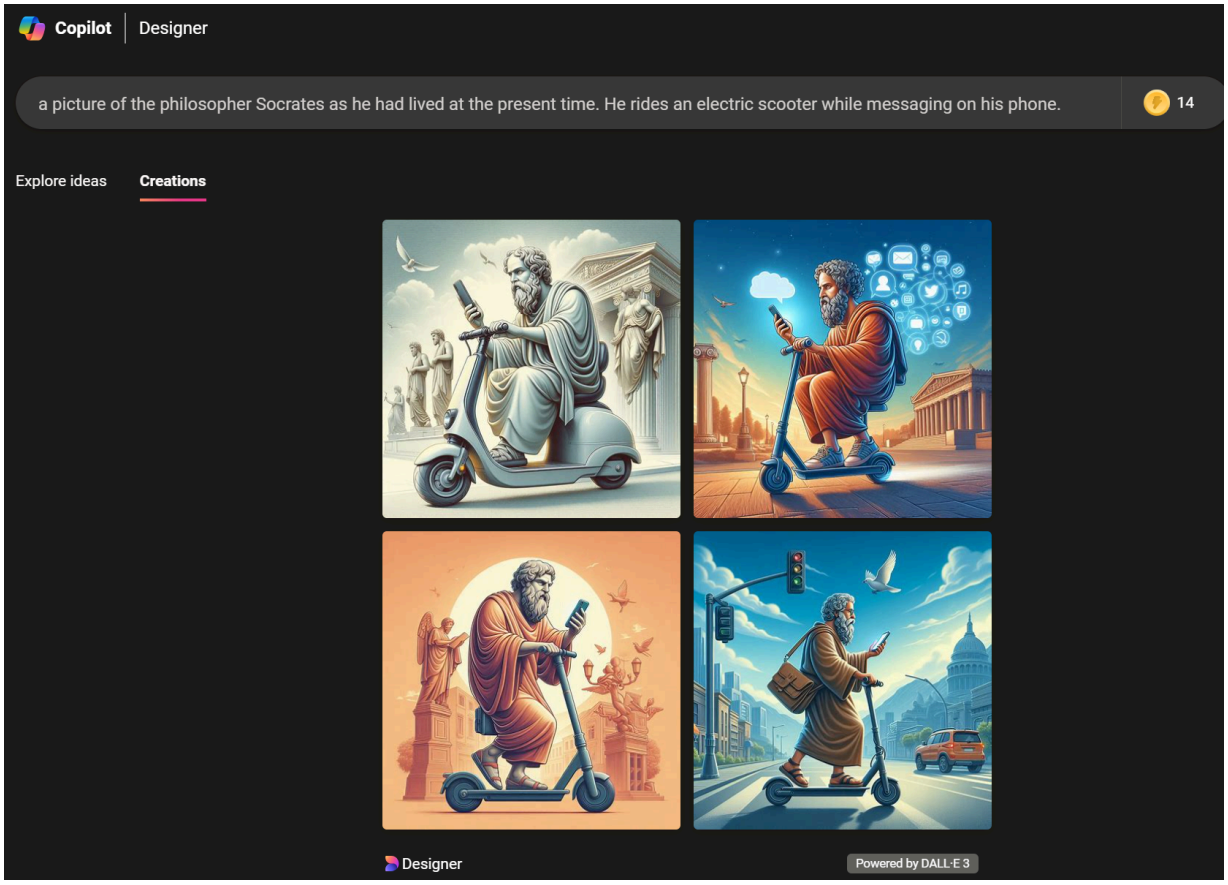


Figure 1.5 – Images generated by DALL-E3 with a natural language prompt as input

It's impressive to see the level of improvement of this model in less than 18 months. Funny enough, we are just scraping the surface of the massive improvements occurring over the last months in the field of Generative AI.

Music generation

The first approaches to generative AI for music generation trace back to the 50s, with research in the field of algorithmic composition, a technique that uses algorithms to generate musical compositions. In fact, in 1957, Lejaren Hiller and Leonard Isaacson created the Illiac Suite for String Quartet

(https://www.youtube.com/watch?v=n0njBFLQSk8), the first piece of music entirely composed by AI. Since then, the field of generative AI for music has been the subject of ongoing research for several decades. Among recent years' developments, new architectures and frameworks have become widespread among the general public, such as the WaveNet architecture introduced by Google in 2016, which has been able to generate high-quality audio samples, or the Magenta project, also developed by Google, which uses **Recurrent Neural Networks** (**RNNs**) and other ML techniques to generate music and other forms of art. Then, in 2020, OpenAI also announced Jukebox, a neural network that generates music, with the possibility to customize the output in terms of musical and vocal style, genre, reference artist, and so on.

Those and other frameworks became the foundations of many AI composer assistants for music generation. An example is Flow Machines, developed by Sony CSL Research. This generative AI system was trained on a large database of musical pieces to create new music in a variety of styles. It was used by French composer Benoît Carré to compose an album called *Hello World* (https://www.helloworldalbum.net/), which features collaborations with several human musicians.

Here, you can see an example of a track generated entirely by Music Transformer, one of the models within the Magenta project:

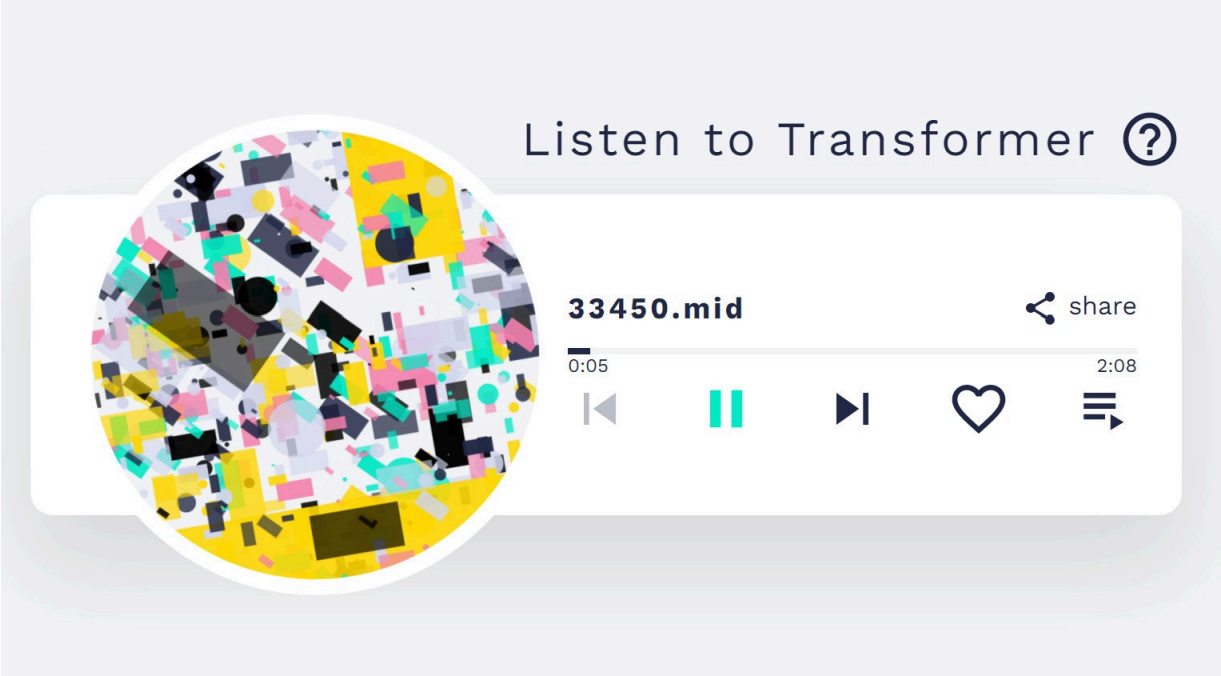


Figure 1.6 – Music Transformer allows users to listen to musical performances generated by AI

Another incredible application of generative AI within the music domain is speech synthesis. It is indeed possible to find many AI tools that can create audio based on text inputs in the voices of well-known singers.

For example, if you have always wondered how your songs would sound if Kanye West performed them, well, you can now fulfill your dreams with tools such as FakeYou.com (https://fakeyou.com/), Deep Fake Text to Speech, or UberDuck.ai (https://uberduck.ai/).

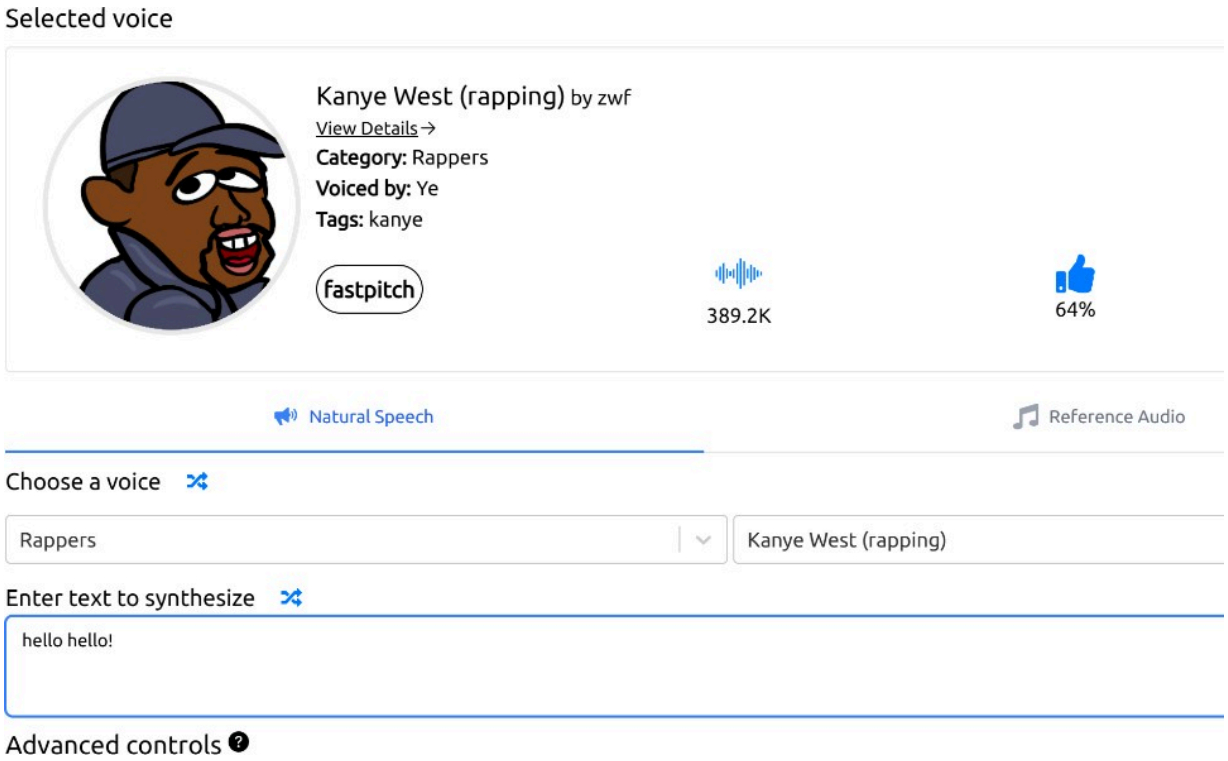
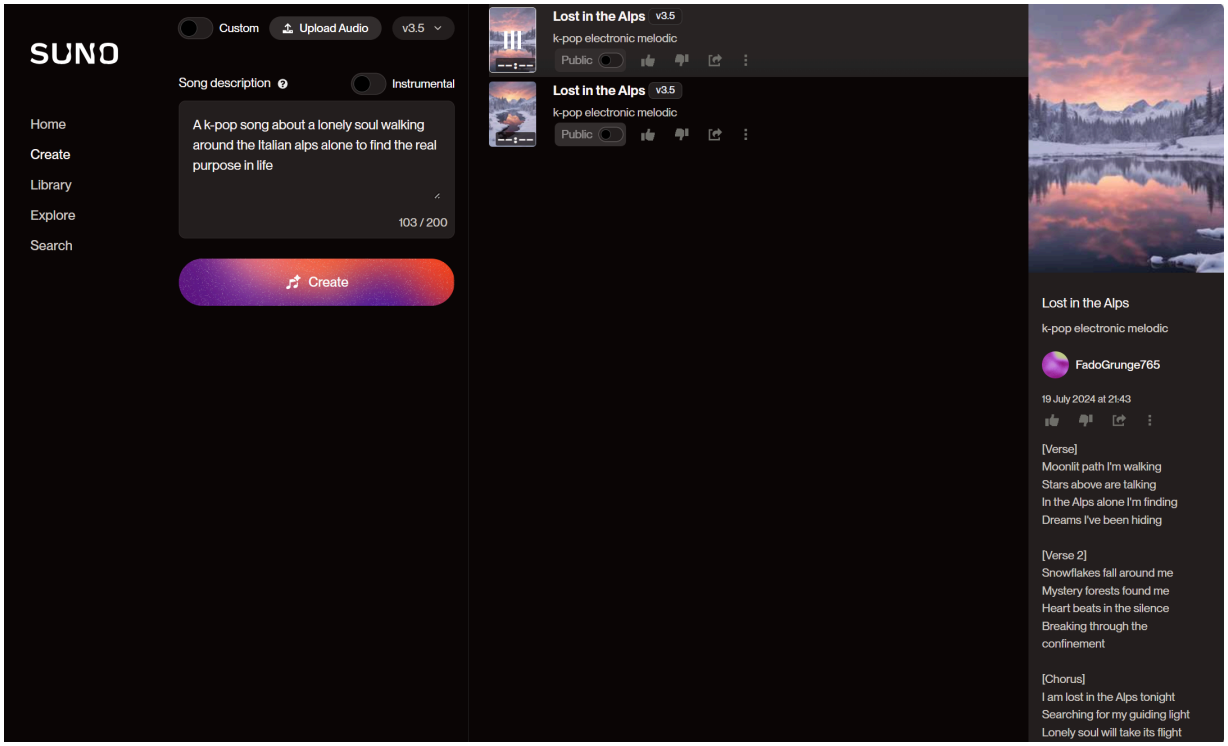


Figure 1.7 – Text-to-speech synthesis with UberDuck.ai

I have to say, the result is really impressive. If you want to have fun, you can also try voices from your all your favourite cartoons as well, such as Winnie The Pooh…

Let's go even further. What if we could generate a song from scratch, just asking the GenAI to do that for us in natural language? Well, today we can do that seamlessly and without any knowledge about music. Among the GenAI products that are rising in the musing market today, a great example is Suno, whose mission is *"[…] is building a future where anyone can make great music. Whether you're a shower singer or a charting artist, we break barriers between you and the song you dream of making. No instrument needed, just imagination. From your mind to music."* (source: https://suno.com/about).



Can you believe that it became my summer 2024 hit? If you want to create your summer hit too, you can try it for free at https://suno.com/create.

Video generation

Generative AI for video generation shares a similar timeline of development with image generation. In fact, one of the key developments in the field of video generation has been the development of GANs. Thanks to their accuracy in producing realistic images, researchers have started to apply these techniques to video generation as well. One of the most notable examples of GAN-based video generation is DeepMind's **Motion to Video**, which generated high-quality videos from a single image.

and a sequence of motions. Another great example is NVIDIA's **Video-to-Video Synthesis** (**Vid2Vid**) DL-based framework, which uses GANs to synthesize high-quality videos from input videos.

The Vid2Vid system can generate temporally consistent videos, meaning that they maintain smooth and realistic motion over time. The technology can be used to perform a variety of video synthesis tasks, such as the following:

- Converting videos from one domain into another (for example, converting a daytime video into a nighttime video or a sketch into a realistic image)
- Modifying existing videos (for example, changing the style or appearance of objects in a video)
- Creating new videos from static images (for example, animating a sequence of still images)

In September 2022, Meta's researchers announced the general availability of **Make-A-Video** (https://makeavideo.studio/), a new AI system that allows users to convert their natural language prompts into video clips. Behind such technology, you can recognize many of the models we mentioned for other domains so far – language understanding for the prompt, image and motion generation with image generation, and background music made by AI composers.

Now, everything we've mentioned above pales in comparison with the latest text-to-video models. To name one, OpenAI announced in February 2024 a new text-to-video model called SORA, releasing some early experiments that were something unexpected:

Prompt: A stylish woman walks down a Tokyo street filled with warm glowing neon and animated city signage. She wears a black leather jacket, a long red dress, and black boots, and carries a black purse. She wears sunglasses and red lipstick. She... +

Prompt: A movie trailer featuring the adventures of the 30 year old space man wearing a red wool knitted motorcycle helmet, blue sky, salt desert, cinematic style, shot on 35mm film, vivid colors.

Figure 1: Videos generated by SORA from a prompt in natural language. Source: https://openai.com/index/sora/

I do encourage you to visit the SORA webpage to have a look at the amazing videos it created. At the time of writing, SORA is not publicly available, as it is going through several tests ran by the OpenAI Red Team.

Overall, generative AI has impacted many domains for years, and some AI tools already consistently support artists, organizations, and general users. The future seems very promising; however, before jumping to the ultimate models available on the market today, we first need to have a deeper understanding of the roots of generative AI, its research history, and the recent developments that eventually lead to the current OpenAI models.

The history and current status of research

In previous sections, we had an overview of the most recent and cutting-edge technologies in the field of generative AI, all developed in recent years. However, the research in this field can be traced back decades ago.

We can mark the beginning of research in the field of generative AI in the 1960s, when Joseph Weizenbaum developed the chatbot ELIZA, one of the first examples of an NLP system. It was a simple rules-based interaction system aimed at entertaining users with responses based on text input, and it paved the way for further developments in both NLP and generative AI. However, we know that modern generative AI is a subfield of DL and, although the first **Artificial Neural Networks** (**ANNs**) were

first introduced in the 1940s, researchers faced several challenges, including limited computing power and a lack of understanding of the biological basis of the brain. As a result, ANNs hadn't gained much attention until the 1980s when, in addition to new hardware and neuroscience developments, the advent of the **backpropagation** algorithm facilitated the training phase of ANNs. Indeed, before the advent of backpropagation, training Neural Networks was difficult because it was not possible to efficiently calculate the gradient of the error with respect to the parameters or weights associated with each neuron, while backpropagation made it possible to automate the training process and enabled the application of ANNs.

Then, by the 2000s and 2010s, the advancement in computational capabilities, together with the huge amount of available data for training, yielded the possibility of making DL more practical and available to the general public, with a consequent boost in research.

In 2013, Kingma and Welling introduced a new model architecture in their paper *Auto-Encoding Variational Bayes*, called **Variational Autoencoders** (**VAEs**). VAEs are generative models that are based on the concept of variational inference. They provide a way of learning with a compact representation of data by encoding it into a lower-dimensional space called **latent space** (with the *encoder* component) and then decoding it back into the original data space (with the *decoder* component).

The key innovation of VAEs is the introduction of a probabilistic interpretation of the latent space. Instead of learning a deterministic mapping of the input to the latent space, the encoder maps the input to a probability distribution over the

latent space. This allows VAEs to generate new samples by sampling from the latent space and decoding the samples into the input space.

For example, let’s say we want to train a VAE that can create new pictures of cats and dogs that look like they could be real.

To do this, the VAE first takes in a picture of a cat or a dog and compresses it down into a smaller set of numbers into the latent space, which represent the most important features of the picture. These numbers are called **latent variables**.

Then, the VAE takes these latent variables and uses them to create a new picture that looks like it could be a real cat or dog picture. This new picture may have some differences from the original pictures, but it should still look like it belongs in the same group of pictures.

The VAE gets better at creating realistic pictures over time by comparing its generated pictures to the real pictures and adjusting its latent variables to make the generated pictures look more like the real ones.

VAEs paved the way toward fast development within the field of generative AI. In fact, only 1 year later, GANs were introduced by Ian Goodfellow. Differently from VAEs architecture, whose main elements are the encoder and the decoder, GANs consist of two Neural Networks – a generator and a discriminator – which work against each other in a zero-sum game.

The generator creates fake data (in the case of images, it creates a new image) that is meant to look like real data (for example, an image of a cat). The discriminator takes in both real and fake

data, and tries to distinguish between them – it's the *critic* in our art forger example.

During training, the generator tries to create data that can fool the discriminator into thinking it's real, while the discriminator tries to become better at distinguishing between real and fake data. The two parts are trained together in a process called **adversarial training**.

Over time, the generator gets better at creating fake data that looks like real data, while the discriminator gets better at distinguishing between real and fake data. Eventually, the generator becomes so good at creating fake data that even the discriminator can't tell the difference between real and fake data.

Here is an example of human faces entirely generated by a GAN:

Figure 1.8 – Examples of photorealistic GAN-generated faces (taken from Progressive Growing of GANs for Improved Quality, Stability, and Variation, 2017: https://arxiv.org/pdf/1710.10196.pdf)

Both models – VAEs and GANs – are meant to generate brand new data that is indistinguishable from original samples, and their architecture has improved since their conception, side by side with the development of new models such as PixelCNNs, proposed by Van den Oord and his team, and WaveNet, developed by Google DeepMind, leading to advances in audio and speech generation.

Another great milestone was achieved in 2017 when a new architecture, called *Transformer*, was introduced by Google

researchers in the paper, – *Attention Is All You Need*, was introduced in a paper by Google researchers. It was revolutionary in the field of language generation since it allowed for parallel processing while retaining memory about the context of language, outperforming the previous attempts of language models founded on RNNs or **Long Short-Term Memory** (**LSTM**) frameworks.

Transformers were indeed the foundations for massive language models called **Bidirectional Encoder Representations from Transformers** (**BERT**), introduced by Google in 2018, and they soon become the baseline in NLP experiments.

Transformers are also the foundations of all the **Generative Pre-Trained** (**GPT**) models introduced by OpenAI, including GPT-3, the model behind ChatGPT, as well as other Large Foundation Models.

Although there was a significant amount of research and achievements in those years, it was not until the second half of 2022 that the general attention of the public shifted toward the field of generative AI.

Not by chance, 2022 has been dubbed the *year of generative AI*. This was the year when powerful AI models and tools became widespread among the general public: diffusion-based image services (MidJourney, DALL-E 2, and Stable Diffusion), OpenAI's ChatGPT, text-to-video (Make-a-Video and Imagen Video), and text-to-3D (DreamFusion, Magic3D, and Get3D) tools were all made available to individual users, sometimes also for free.

This had a disruptive impact for two main reasons:

- Once generative AI models have been widespread to the public, every individual user or organization had the possibility to experiment with and appreciate its potential, even without being a data scientist or ML engineer.
- The output of those new models and their embedded creativity were objectively stunning and often concerning. An urgent call for adaptation—both for individuals and governments—rose.

Henceforth, in the very near future, we will probably witness a spike in the adoption of AI systems for both individual usage and enterprise-level projects.

18 months later: main trends and innovations

Fast forward from November 2022 to today, we have witnessed a huge amount of innovation in the field of GenAI. Many of these innovations are linked to the brand-new models developed and released to the public, like the OpenAI's GPT-4o and DALL-E3, but also Google Gemini, Meta Llama 3, Microsoft Phi3 and many others.

However, the most remarkable achievements probably lie in the way we interact with and build applications around those models. In this section, we are going to explore three main advancements that have marked the most popular reference architectures for GenAI-powered applications.

Retrieval Augmented Generation

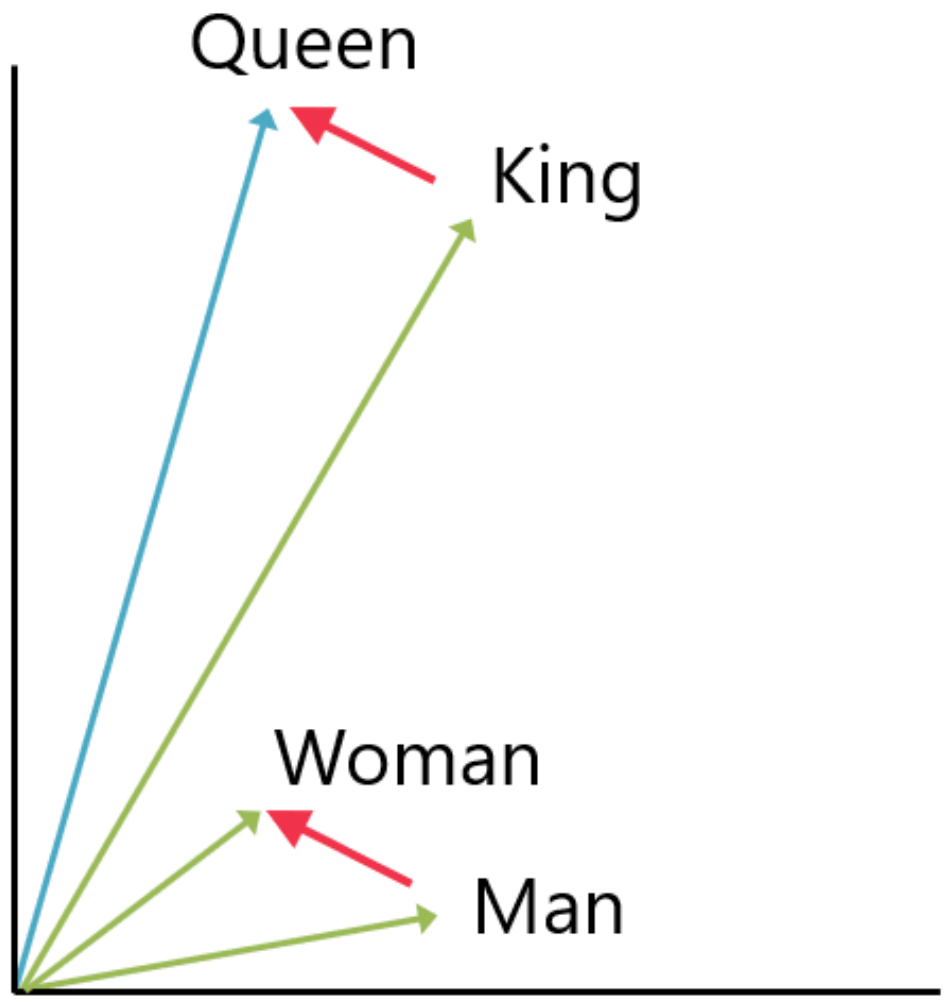
One of the first limitations of ChatGPT and, generally speaking, of LLMs, was that of the knowledge base cutoff. In fact, the knowledge of LLMs is limited to the training set (made od public information from the internet) they have been trained on and, as long as this can be exhaustive, it's not up to date. Plus, it is missing the proprietary knowledge base that might be relevant for us or our organization. For example, if you ask ChatGPT "what are my company's policy for employees health insurance?", the model won't be able to answer since it has no access to this information.

To bypass this limitation, a new framework was designed to allow LLMs to navigate through customized documentations that we can provide. This framework is called Retrieval Augmented Generation (RAG).

The idea behind RAG is that of decoupling the LLM from the knowledge base we want to navigate through. To make it possible, the external knowledge base needs to be transformed into numerical vectors through a process called embedding and stored into a specialized Vector Database.

> An embedding is a way of representing high-dimensional, non-numeric data, such as words or sentences, in a lower-dimensional space, such as vector. A text embedding can capture the semantic and syntactic features of the text, such as meaning, context, and similarity.
>
> > Each embedding is a vector of floating-point numbers, such that the distance between two embeddings in the vector space is correlated with semantic similarity between two inputs
> > in the original format.

For example, if two concepts are similar, then their vector representations should also be similar.

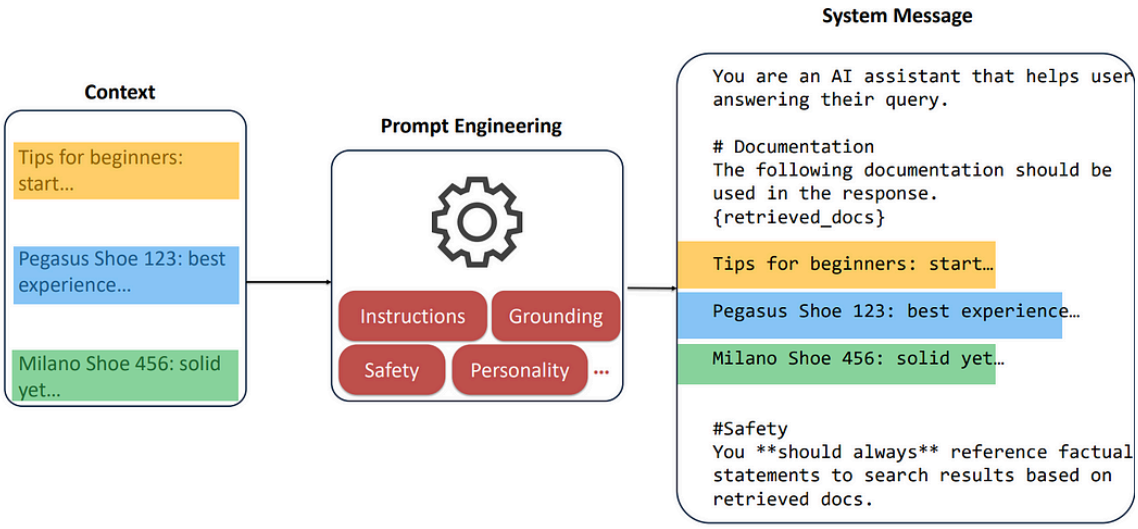


King-Man+Woman ≈ Queen

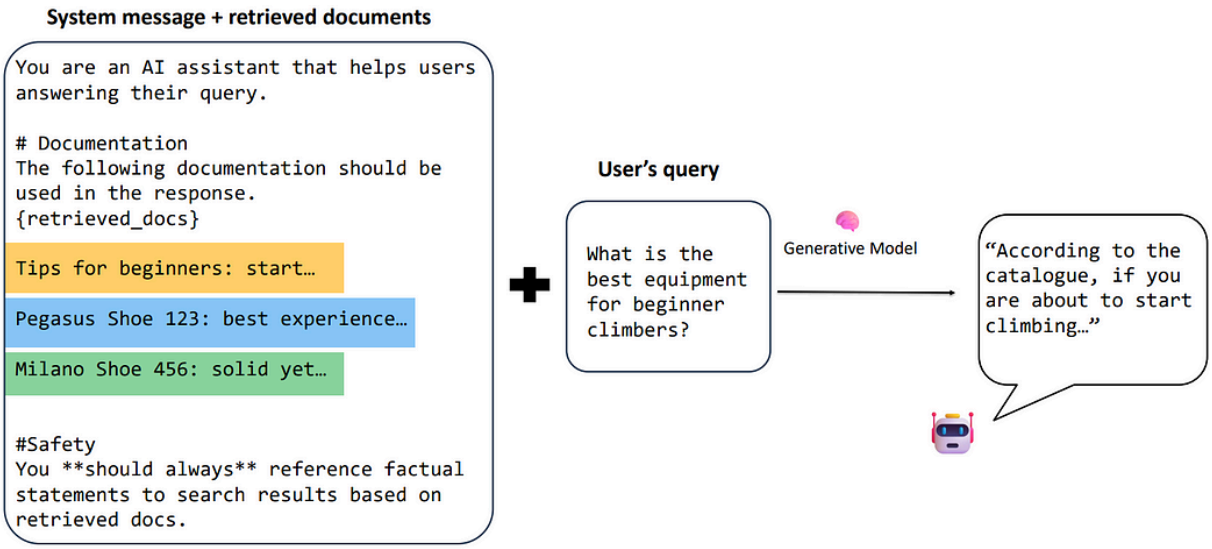
RAG is made of three phases:

- **Retrieval**: given a user's query and its corresponding vector, the most similar pieces of documents (those corresponding to the vectors that are closer to the user query's vector) are retrieved and used as the base context for the LLM.

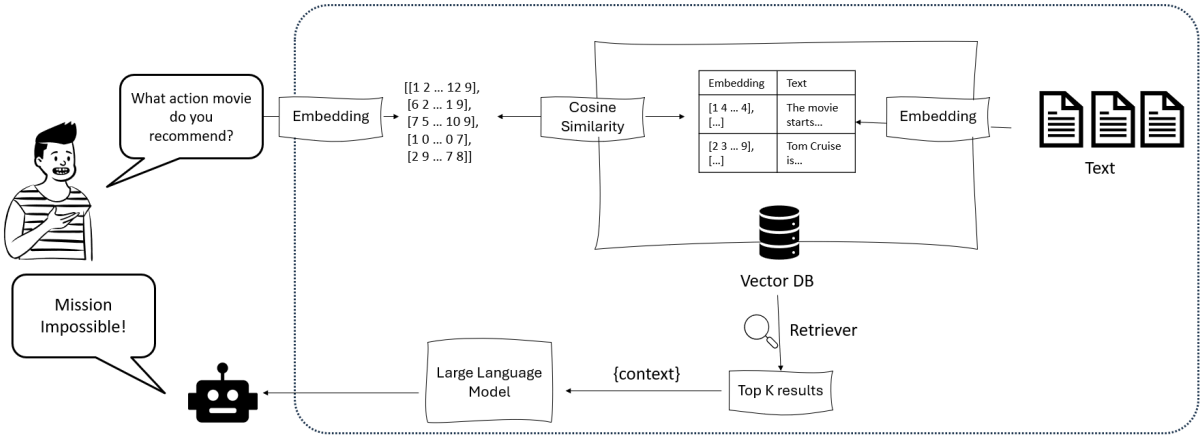
- **Augmentation**: the retrieved context is enriched through additional instructions, rules, safety guardrails and similar practices that are typical of prompt engineering techniques (we will cover the topic of prompt engineering in Chapter 3).



- **Generation**: based on the augmented context, the LLM generates the response to the user's query.



The overall process looks as follows:



RAG combines the strengths of generative models and information retrieval systems to enhance the quality and relevance of generated content. Traditional generative models rely solely on their training data to produce responses, which

can sometimes result in outdated or irrelevant information. RAG addresses this limitation by integrating external knowledge bases during the generation process.

For example, OpenAI's ChatGPT, when enhanced with RAG, can pull in current data from reliable sources to answer a query about recent events, which would be beyond the scope of its training cut-off. This synergy of retrieval and generation broadens the applicability of generative AI, making it more robust and useful in dynamic, information-rich environments.

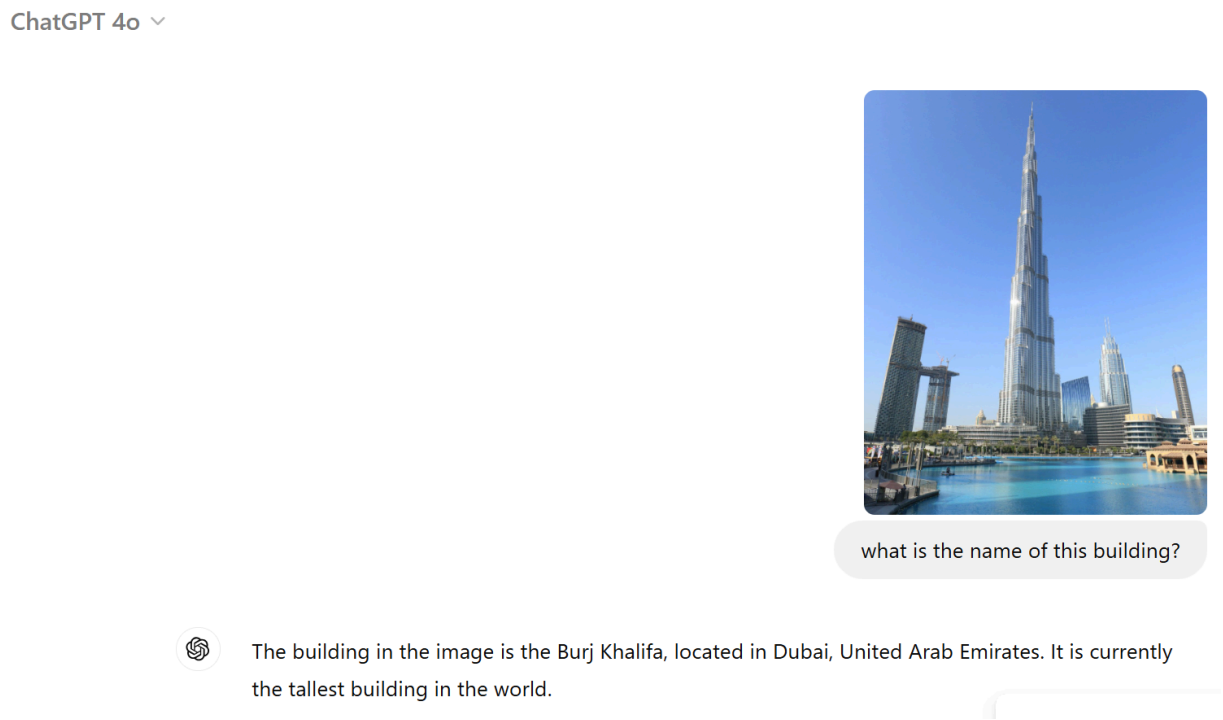
Multimodality

In the first paragraph of this Chapter, we saw the various domains of Generative AI, ranging from text to images, from videos to music. Typically, Large Foundation Models tend to me domain-specific, as we saw for Large Language Models in the case of language understanding and generation, or DALL-E3 in case of image generation.

However, the recent advances in Generative AI have enabled the development of Large Multimodal Models (LMMs) that can process and generate different types of data, such as text, images, audio, and video.

LMMs share with “standard” Large Language Models (LLMs) the capability of generalization and adaptation typical of Large Foundation Models. However, LMMs are capable of processing heterogeneous data with the idea of mirroring the way humans interact with the surrounding ecosystem – that is, with all our senses.

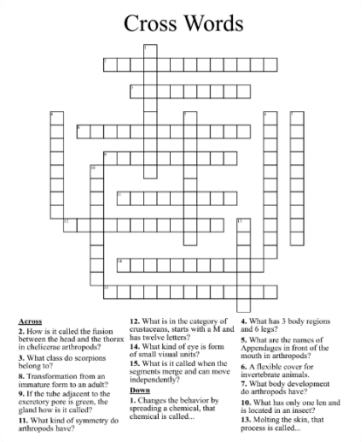
A great example of multimodal model is OpenAI's GPT-4o, which is able to interact with users via text, images and audio. Let's see the following example:



As you can see, the model was able to analyze the image and reason over it. Let's now go ahead and ask the model to generate an illustration:

thanks. Can you generate a cartoon style illustration of it? black and white

The most interesting fact about LMMs is that they preserve their reasoning capabilities, making them suitable for complex reasoning in heterogeneous data contexts. Let’s consider this last example (showing only the first lines of the response):



can you solve this for me?

Sure, let's solve the crossword puzzle. Here are the answers based on the given clues:

Across:

2. **Cephalothorax** (Fusion between the head and thorax in chelicerate arthropods)
3. **Arachnids** (Class that scorpions belong to)
4. **Metamorphosis** (Transformation from an immature form to an adult)

As you may imagine, this opens a landscape of applications in various industries, and we are going to see some concrete examples in the upcoming chapters.

AI Agents

In previous sections, we uncovered how LFMs are great when it comes to resonate and generate content. However, they lack one ability, that is the one of taking actions and interact with the surrounding ecosystem that goes beyond the single user. For example, what if we want our LLM to be able not only to generate an amazing LinkedIn post, but also publish it on our page?

To overcome this limitation, AI agents emerge as key players. But what exactly are they? Agents can be seen as AI systems powered by Large Language Models that, given a user's query, are able to interact with the surrounding ecosystem to the extent to which we allow them to. The perimeter of the ecosystem is delimited by the tools (or plug-ins) we provide the agents with (in our previous example, we might provide the agent with a LinkedIn plug-in, so that it is able to post the generated content)

Agents are made of the following ingredients:

- An LLM which acts as the reasoning engine of the AI system
- A system message which instructs the Agent to behave and think in a given way. For example, you can design an Agent as a teaching assistant for students with the following system message: "You are a teaching assistant. Given a student's query, NEVER provide the final answer, but rather provide some hints to get there".
- A set of tools the Agent can leverage to interact with the surrounding ecosystem.

AI Agents are a perfect representation of the meaning of "LLM as reasoning engine of an application". In fact, the beauty of Agents is that they can pick the best tool to use to accomplish a user's request. For example, let's say we have an AI agent to produce LinkedIn content, and we provide it with two tools: a LinkedIn plug-in and a web search plug-in (each one with a proper description of its functionality). Then, let's explore the behaviour of the agent upon two different questions:

- Generate a story about a little dog walking around the mountains→the Agent will generate the story without

involving any plug-in.
- Generate a story about the current weather in Milan→the Agent will invoke the web search plug-in to get the current weather in Milan.
- Generate a LinkedIn post about the current weather in Mialn and publish it on my profile:the Agent will invoke the web search plug-in to get the current weather in Milan and the LinkedIn plug-in to post it on my profile.

The combinations of instructions and set of plug-ins make AI Agents extremely versatile, and you can create highly specialized entities to address specific scenarios.

And that’s not all.

Why having only one Agents if you can create your own crew of Agents talking to and cooperating among each other? Imagine multiple agents, each one with a specific expertise and goal, communicating and interacting among each other to accomplish a task. This is how **multi-agent applications** look like, and in the last few months, this pattern started showing emergent behaviors.

Let’s consider the following example. We want to generate an elevator pitch about the climate change. We need up to date information to do so (latest trends and research, future perspectives and so on), as well as solid research grounded by academic papers. Plus, we need to be concise yet sharp and effective, delivering all the key information in a very short pitch.

Now, we could ask all of that to a single agent, providing it with all the needed tools and with long instructions to accomplish the task. However, it is proven that LLMs tend to perform worse

when we give them “too many things to do”. Instead, let’s use a multi-agent approach, creating a team with the following AI professionals:

- A market analyst who can search the web for the latest news about climate change this will be an Agent with a web search plug-in and specific instructions to search for news;
- An expert researcher who can easily navigate through academic research papers about climate change this will be an Agent with an Arxiv plug-in and specific instructions on how to retrieve relevant information;
- A expert in public speaking who can easily consolidate all the information in one elevator pitch this will be an Agent with proper instructions on how to deliver perfect pitches;
- A critic who will review the pitch and propose some changes to the expert in public speaking, if needed this will be an Agent with proper instructions on how to deliver perfect pitches;

So, upon the user’s question “Generate an elevator pitch about the current issue of climate change”, all the agents can start working on the project.

There are many frameworks that can help developer with multi-agent applications (including AutoGen, LangGraph, CrewAI), especially when it comes to the “flow” that we want our agents to follow. For example, we might want to enforce a specific number of iterations; or that all agents are invoked at least once; or even to involve us, as users, every iteration to provide further feedback to be incorporated in the upcoming iteration.

At the time of writing, multi-agent framework is showing promising advancements, yet it is still in an experimentation

phase and probably still far from being enterprise-ready. Nevertheless, it is a glimpse of the outstanding reasoning capabilities behind LLMs and how they can unlock new ways of problem solving and creation.

Small Language Models

Large Language Models are, unsurprisingly, large. It means that the architecture of the Artificial Neural Network featuring LLMs is made of a huge amount of parameters, of the order of trillions. Ideally, the higher the number of parameters, the better the reasoning capabilities of the LLM. However, with high numbers of parameters come high cost of training and hosting, since a powerful AI infrastructure is needed. Plus, the energy consumption required by those models raise serious questions about the environmental impact of LLMs training and their overall sustainability in the long run. To give you an idea, these are some numbers related to open-source LLMs with different sizes (measured in number of parameters):

	GPU Type	GPU Power consumption	GPU-hours	Total power consumption	Carbon emitted (tCO_2eq)
OPT-175B	A100-80GB	400W	809,472	356 MWh	137
BLOOM-175B	A100-80GB	400W	1,082,880	475 MWh	183
LLaMA-7B	A100-80GB	400W	82,432	36 MWh	14
LLaMA-13B	A100-80GB	400W	135,168	59 MWh	23
LLaMA-33B	A100-80GB	400W	530,432	233 MWh	90
LLaMA-65B	A100-80GB	400W	1,022,362	449 MWh	173

Figure 2: Carbon footprint and compute power required to train different models in the same datacenter. Source: https://arxiv.org/pdf/2302.13971

Luckily, over the last months, we witnessed a raising interest and investment in the research of smaller models, with the goal of maintaining decent reasoning capabilities while reducing the number of parameters. The idea is that smaller models, even if “less intelligent”, can still reach the desired requirements if specialized in more tailored activities. After all, do we really need huge and super powerful models of trillions of parameter to solve all the tasks that we have in mind?

These smaller models are called Small Language Models (SMLs) and, beside being lighter and less demanding in terms of infrastructure, they are also showing surprisingly high performance.

For example, if we consider the smaller version of Phi-3, part of the popular SLM Phi family developed by Microsoft, with “only” 7 billion parameters, we can see that it beats GPT-3.5-turbo in all the most popular LLMs benchmarks:

Category	Benchmark	Phi-3-Small		Gemma-7b	Mixtral-8x7b	Llama-3-8B-In	GPT3.5-Turbo-1106
		Phi-3-Small-8K-In	Phi-3-Small-128K-In				
Popular Aggregate Benchmarks	MMLU (5-shot)	75.7	75.5	63.6	70.5	66.5	71.4
Language Understanding	HellaSwag (5-shot)	77.0	79.6	49.8	70.4	71.1	78.8
Reasoning	WinoGrande (5-shot)	81.5	80.1	55.6	62	65	68.8
	Social IQA (5-shot)	79.2	78.7	65.5	75.9	73.9	68.3
	TruthfulQA (MC2) (10-shot)	70.2	69.6	52.1	60.1	63.2	67.7
	MedQA (2-shot)	65.4	46.3	49.6	62.2	60.5	63.4
Factual Knowledge	TriviaQA (5-shot)	58.1	66.0	72.3	82.2	67.7	85.8
Math	GSM8K CoT (8-shot)	89.6	87.3	59.8	64.7	77.4	78.1
Code generation	HumanEval (0-shot)	61.0	59.1	34.1	37.8	60.4	62.2
	MBPP (3-shot)	71.7	70.3	51.5	60.2	67.7	77.8

Figure 3: benchmark table of Phi-3 Small capabilities compared to other SLMs and LLMs. Source: https://azure.microsoft.com/en-us/blog/introducing-phi-3-redefining-whats-possible-with-slms/

Now, we might think that GPT-3.5-turbo is kind of "deprecated", however we have to remember that, with its 175B parameters, it used to be the most powerful model in the market just 1 year ago, and it is remarkable to see that a 7B model is capable of better results.

That of SLMs is definitely a research stream to keep an eye on, especially when it comes to those scenarios where I might want to deploy my model locally or even customizing it with fine-tuning (we will cover fine tuning in the next chapter).

Summary

In this chapter, we explored the exciting world of generative AI and its various domains of application, including image generation, text generation, music generation, and video generation. We learned how generative AI models such as ChatGPT and DALL-E, trained by OpenAI, use DL techniques to learn patterns in large datasets and generate new content that is both novel and coherent. We also discussed the history of generative AI, its origins, and the current status of research on it.

The goal of this chapter was to provide a solid foundation in the basics of generative AI and to inspire you to explore this fascinating field further.

In the next chapter, we will focus on one of the most promising technologies available on the market today, ChatGPT: we will go

through the research behind it and its development by OpenAI, the architecture of its model, and the main use cases it can address as of today.

References

- https://arxiv.org/abs/1406.2661
- https://www.youtube.com/watch?v=Iy9vRvyRf_E
- https://arxiv.org/abs/1912.04958
- This person does not exist: https://this-person-does-not-exist.com
- https://arxiv.org/abs/1808.06601
- https://www.microsoft.com/en-us/research/blog/a-deep-generative- model-trifecta-three-advances-that-work-towards-harnessing- large-scale-power/
- https://tcwang0509.github.io/vid2vid/
- https://arxiv.org/pdf/2302.13971
- https://azure.microsoft.com/en-us/blog/introducing-phi-3-redefining-whats-possible-with-slms/

2 OpenAI and ChatGPT: Beyond the Market Hype

Join our book community on Discord

https://packt.link/EarlyAccess

This chapter provides an overview of OpenAI and its most notable development—ChatGPT, highlighting its history, technology, and capabilities.

The overall goal is to provide a deeper knowledge of how ChatGPT can be used in various industries and applications to improve communication and automate processes and, finally, how those applications can impact the world of technology and beyond.

We will cover all this with the following topics:

- What is OpenAI?
- Overview of OpenAI model families
- Road to ChatGPT: the math of the model behind it
- ChatGPT: the state of the art

Technical requirements

In order to be able to test the example in this chapter, you will need the following:

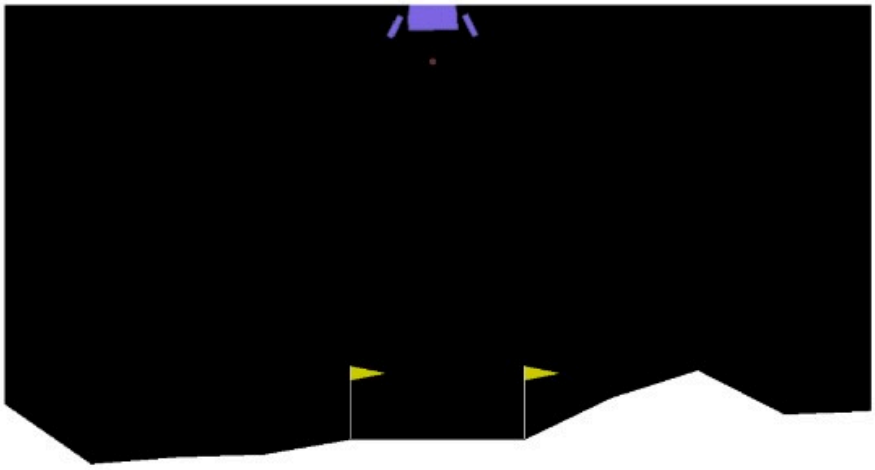
- An OpenAI account to access the Playground and the Models API (https://openai.com/api/login20)
- Your favorite IDE environment, such as Jupyter or Visual Studio
- Python 3.7.1+ installed (https://www.python.org/downloads)
- `pip` installed (https://pip.pypa.io/en/stable/installation/)
- OpenAI Python library (https://pypi.org/project/openai/)

What is OpenAI?

OpenAI is a research organization founded in 2015 by Elon Musk, Sam Altman, Greg Brockman, Ilya Sutskever, Wojciech Zaremba, and John Schulman. As stated on the OpenAI web page, its mission is “to ensure that Artificial General Intelligence (AGI) benefits all of humanity”. As it is general, AGI is intended to have the ability to learn and perform a wide range of tasks, without the need for task-specific programming.

Since 2015, OpenAI has focused its research on **Deep Reinforcement Learning** (**DRL**), a subset of **machine learning** (**ML**) that combines **Reinforcement Learning** (**RL**) with deep neural networks. The first contribution in that field traces back to 2016 when the company released OpenAI Gym, a toolkit for researchers to develop and test **RL** algorithms.

Gym is a standard API for reinforcement learning, and a diverse collection of reference environments



The Gym interface is simple, pythonic, and capable of representing general RL problems:

```
import gym
env = gym.make("LunarLander-v2", render_mode="human")
observation, info = env.reset(seed=42)
for _ in range(1000):
   action = policy(observation)  # User-defined policy function
   observation, reward, terminated, truncated, info = env.step(action)

   if terminated or truncated:
      observation, info = env.reset()
env.close()
```

Figure 2.1 – Landing page of Gym documentation (https://www.gymlibrary.dev/)

OpenAI kept researching and contributing in that field, yet its most notable achievements are related to generative models—**Generative Pre-trained Transformers** (**GPT**).

After introducing the model architecture in their paper “Improving Language Understanding by Generative Pre-Training” and baptizing it **GPT-1**, OpenAI researchers soon released, in 2019, its successor, the GPT-2. This version of the GPT was trained on a corpus called **WebText**, which at the time contained slightly over 8 million documents with a total of 40 GB of text from URLs shared in Reddit submissions with at least 3 upvotes. It had 1.2 billion parameters, ten times as many as its predecessor.

Here, you can see the landing page of a UI of GPT-2 published by HuggingFace ([https:// transformer.huggingface.co/doc/distil-gpt2](https://transformer.huggingface.co/doc/distil-gpt2)):

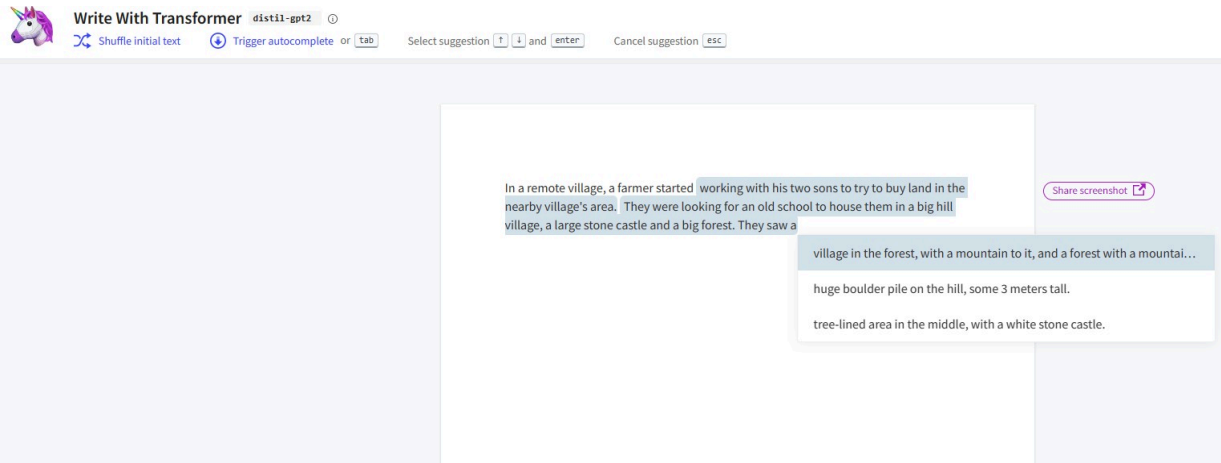


Figure 2.2 – GPT-2 writing a paragraph based on a prompt. Source: https://transformer.huggingface.co/doc/distil-gpt2

Then, in 2020, OpenAI first announced and then released GPT-3, which, with its 175 billion parameters, dramatically improved benchmark results over GPT-2. It was with the GPT-3 model – more precisely, with its fine-tuned version called GPT-3.5 – that we entered the era of ChatGPT: in fact, at its first release, ChatGPT was powered by GPT-3.5.

From that moment (November 2022) to today, OpenAI released many new version of its GPT series: GPT-4, GPT-4-turbo, GPT-4vision

(first multimodal model) and GPT-4o, where the “o” stands for “omni”, referring to its multimodal capabilities.

In addition to natural language generative models, OpenAI also developed in other fields of Generative AI, including:

- **DALL-E**→ A text-to-image model that generates detailed and creative images from textual descriptions. It can produce a wide variety of images, from realistic scenes to imaginative and abstract art.
- **Whisper**→ A speech recognition model designed to transcribe spoken language into text. It excels in understanding various accents, dialects, and noisy environments, making it highly effective for real-time transcription and voice-controlled applications.
- **SORA**→A text-to-video model that generates realistic and imaginative videos from text instructions. SORA can create videos up to a minute long, maintaining visual quality and adherence to the user’s prompt. At the time of writing, SORA is not available to the general public yet.

Although OpenAI has invested in many fields of Generative AI, its contribution to text understanding and generation has been outstanding, thanks to the development of the foundation GPT models we are going to explore in the following paragraphs.

An overview of OpenAI Playground

Today, OpenAI offers a set of pre-trained, ready-to-use models that can be consumed by the general public. This has two important implications:

- Powerful foundation models can be consumed without the need for long and expensive training

- It’s not necessary to be a data scientist or an ML engineer to manipulate those models

Users can test OpenAI models in two ways:

- OpenAI Playground, a friendly user interface where you can interact with models without the need to write any code.
- Via REST API with a pro-code approach. In fact, all OpenAI models – and, generally speaking, LLMs – comes as pre-built components that can be consumed by applications via endpoints and keys.

In this chapter, we will explore the first approach, while we will see how to incorporate OpenAI models’ APIs in Chapter 11.

Before jumping into the Playground, let’s first have an overview of OpenAI’s models families.

OpenAI Models Families

Over the last years, OpenAI made huge advancements in the field of model developments, releasing newer models’ versions at insane speed. In this section, we will see the main models divided by domain.

- **Language Models**→OpenAI's GPTs are advanced language models designed to generate human-like text based on given prompts. They are versatile and can be used for various natural language processing tasks such as text completion, translation, summarization, and coding. This is the field where OpenAI demonstrates outstanding performance thanks to its flagship models family: the GPTs. Since November 2022, when ChatGPT was launched, OpenAI released:

    - GPT-3.5-turbo, the model behind the first version of ChatGPT
    - GPT-4 (first multimodal model) and GPT-4-turbo (optimized for chats and assistants)
    - GPT-4o, the most advanced multimodal model
    - GPT-4o mini, the smaller version of the GPT-4o which is faster and cheaper.

In addition to those latest models, users can access a set of so called “GPT base models”, like the Babbage-002 or davinci-002 (the GPT-2), that represents the legacy completions APIs (we will cover completions APIs in the next section).

- **Image model**→ OpenAI's image models, like DALL-E, are designed to generate and manipulate images from textual descriptions. DALL-E models can create highly detailed and imaginative visuals, enabling users to produce unique artwork, design concepts, and more. For instance, DALL-E 3 (the latest version of the model) is capable of creating intricate and creative images based on complex prompts, pushing the boundaries of what AI can do in the field of visual arts. These models are particularly useful in creative industries, digital marketing, and any field that benefits from high-quality, custom visuals.
- **Text to speech and Speech to text**→ In Chapter 1, we already mentioned the Speech-to-text (STT) model developed by OpenAI, Whisper, that can handle and process audio input. In addition to that, OpenAI also developed Text-to-Speech (TTS) models that provide high-quality outputs that are both intelligible and expressive, making them suitable for a wide range of applications from customer service bots to educational tools. Text-to-Speech (TTS) models developed by OpenAI convert written text into spoken language, producing realistic and natural-sounding audio. These models are essential for

creating voice assistants, improving accessibility for visually impaired users, and generating automated announcements.

- **Text to video**→with the announcement of SORA, OpenAI revealed its cutting-edge text-to-video model capable of generating realistic and imaginative video scenes from text descriptions. By adapting techniques from DALL-E and incorporating transformers, SORA can create high-fidelity videos up to one minute long. It excels in maintaining 3D consistency, object permanence, and simulating interactions within videos. While still facing challenges like accurately modeling complex physics and ensuring temporal coherence, SORA holds significant potential for creative industries, offering new possibilities in video production and storytelling
- **Embeddings**→Embeddings models from OpenAI transform text into numerical representations called vectors (or embeddings) that capture semantic meaning and are projected in a multi-dimensional vector space.

The mathematical distances between different instances in this space represent their similarity in terms of meaning. As an example, imagine the words queen, woman, king, and man. Ideally, in our multidimensional space, where words are vectors, if the representation is correct, we want to achieve the following:

$$\overrightarrow{king} + (\overrightarrow{woman} - \overrightarrow{man}) \approx \overrightarrow{queen}$$

Figure 2.10 – Example of vectorial equations among words

This means that the distance between woman and man should be equal to the distance between Queen and King.

Embeddings can be extremely useful in intelligent search scenarios. Indeed, by getting the embedding of the user input and the documents the user wants to search, it is possible to compute distance metrics (namely, cosine similarity) between the input and the documents. By doing so, we can retrieve the documents that are closer, in mathematical distance terms, to the user input.

The OpenAI's embedding models **text-embedding-ada-002** and **text-embedding-3-large** models offer state-of-the-art performance in tasks such as text similarity, text search, and code search by creating dense vector representations of text. These embeddings allow for efficient and effective comparison of large text datasets, improving search accuracy and relevance

- **Moderation**→OpenAI's moderation models are designed to detect and filter out inappropriate, harmful, or unsafe content in text. These models are integral to maintaining safe and respectful online environments by identifying potentially offensive or harmful language. The latest moderation model, **text-moderation-007**, is robust and effective in ensuring content compliance and safety across platforms. It helps developers and companies enforce community guidelines and prevent the spread of harmful content, thereby promoting safer digital interactions.

Overall, OpenAI has reached more and more coverage over the last years, addressing diverse areas of Generative AI with state-of-the-art models.

Trying OpenAI models in the Playground

To access your OpenAI playground, you need to create an OpenAI account and navigate through

https://platform.openai.com/playground. This is how the landing page looks like:

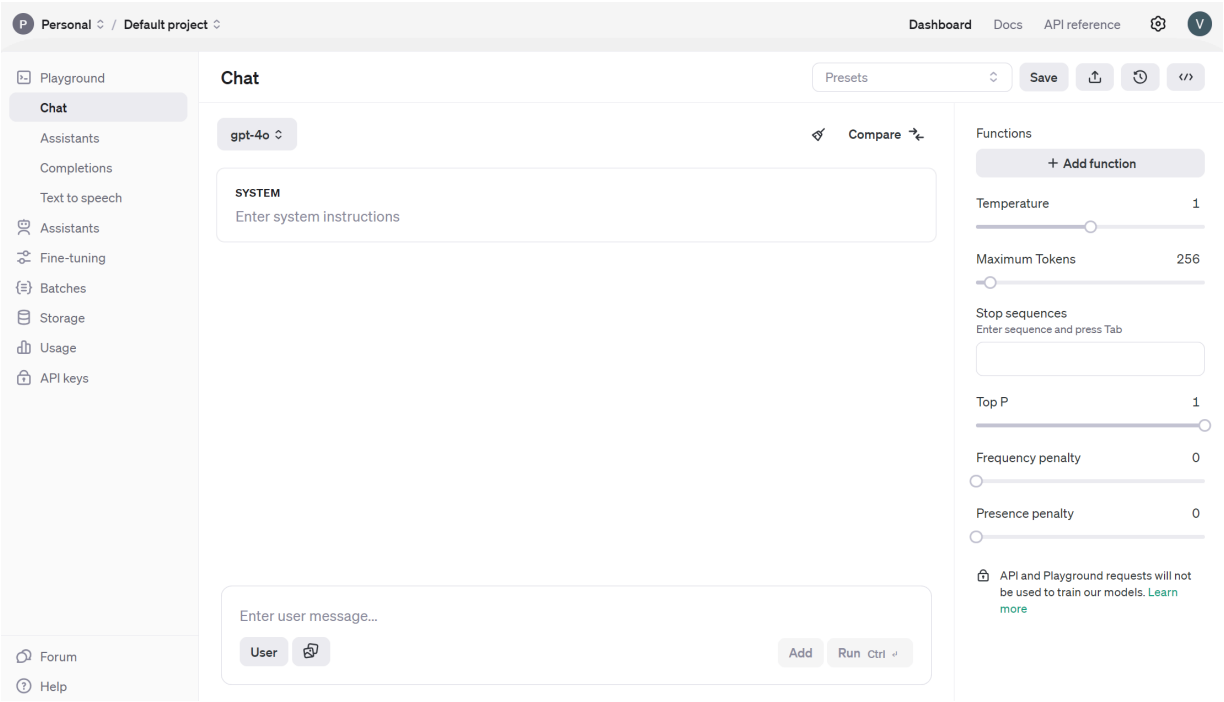


Figure 2.4 – OpenAI Playground at https://platform.openai.com/playground

As you can see from *Figure 2.4*, the Playground offers a UI where the user can start interacting with the model, which you can select on the right-hand side of the UI.

Before diving deeper into the main sections of the playground, let's first define some jargon you will see in this chapter:

- **Tokens**: Tokens can be considered as word fragments or segments that are used by the API to process input prompts. Unlike complete words, tokens may contain trailing spaces or even partial sub-words. To better understand the concept of tokens in terms of length, there are some general guidelines to keep in mind. For instance, one token in English is

approximately equivalent to four characters, or three-quarters of a word.

- **Prompt**: In the context of **natural language processing** (**NLP**) and ML, a prompt refers to a piece of text that is given as input to an AI language model to generate a response or output. The prompt can be a question, a statement, or a sentence, and it is used to provide context and direction to the language model.
- **Context**: In the field of GPT, context refers to the words and sentences that come before the user's prompt. This context is used by the language model to generate the most probable next word or phrase, based on the patterns and relationships found in the training data.
- **Model confidence**: Model confidence refers to the level of certainty or probability that an AI model assigns to a particular prediction or output. In the context of NLP, model confidence is often used to indicate how confident the AI model is in the correctness or relevance of its generated response to a given input prompt.
- **Functions**. Functions is yet another definition for the concept of tools or plug-ins introduced in Chapter 1. With functions, we provide the model with an extra skill that it can invoke to accomplish user's task. A function will always have a description in natural language, so that the model knows when to invoke it.

The preceding definitions will be pivotal in understanding how to use Azure OpenAI model families and how to configure their parameters.

In the Playground, there are four main sections to interact with the models:

Chat: Here you can test all the chat models available today, including both text-only models (like the GPT-3.5) or multimodal models (like the GPT-4o). You can provide a system message – that

is the set of instructions that you provide your model with, all in natural language. You can also compare the output of two different models, given the same user's question. The following is an example on how to do that:

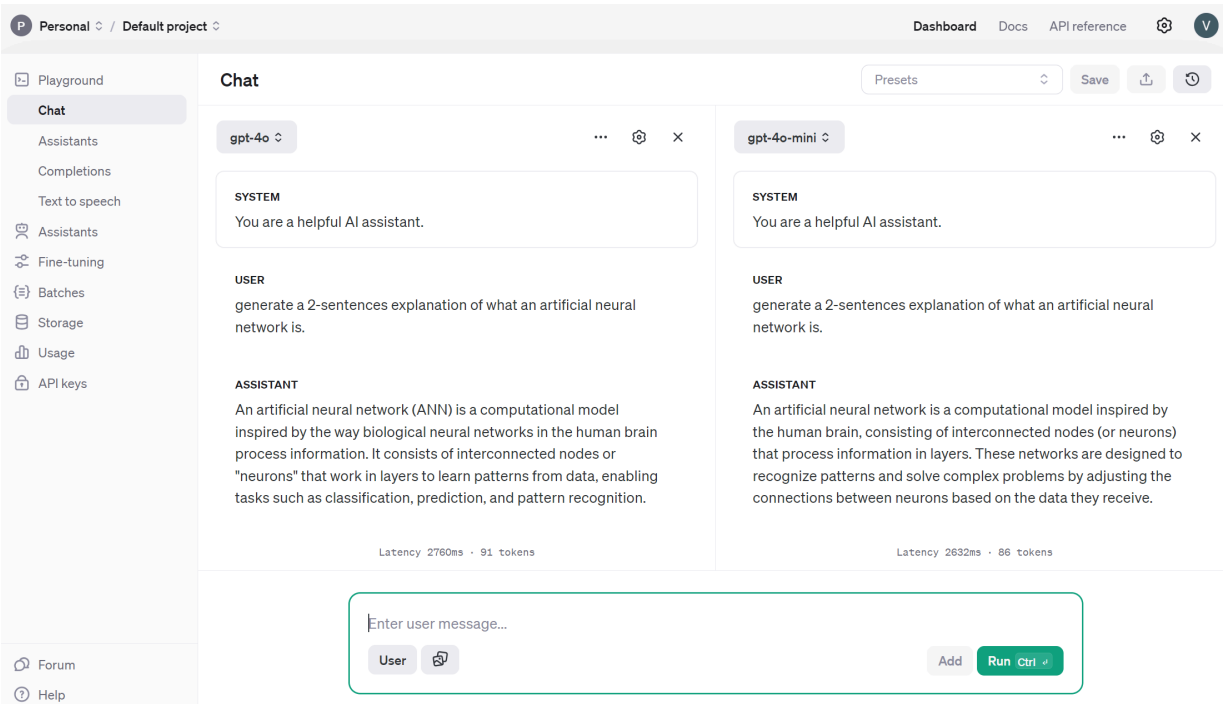


Figure 2.5 – An example of comparison between two models.

For each model, you can also play with some parameters that you can configure. Here is a list:

- **Temperature** (ranging from 0 to 1): This controls the randomness of the model's response. A low-level temperature makes your model more deterministic, meaning that it will tend to give the same output to the same question. For example, if I ask my model multiple times *What is OpenAI?* with temperature set as 0, it will always give the same answer. On the other hand, if I do the same with a model with temperature set as 1, it will try to modify its answers each time, in terms of wording and style.

- **Max tokens:** This controls the length (in terms of tokens) of the model's response to the user's prompt.
- **Stop sequences** (user input): This makes responses end at the desired point, such as the end of a sentence or list.
- **Top probabilities** (ranging from 0 to 1): This controls which tokens the model will consider when generating a response. Setting this to 0.9 will consider the top 90% most likely of all possible tokens. One could ask *Why not set top probabilities as 1 so that all the most likely tokens are chosen?* The answer is that users might still want to maintain variety when the model has low confidence, even in the highest-scoring tokens.
- **Frequency penalty** (ranging from 0 to 1): This controls the repetition of the same tokens in the generated response. The higher the penalty, the lower the probability of seeing the same tokens more than once in the same response. The penalty reduces the chance proportionally, based on how often a token has appeared in the text so far (this is the key difference from the following parameter).
- **Presence penalty** (ranging from 0 to 2): This is similar to the previous one but stricter. It reduces the chance of repeating any token that has appeared in the text at all so far. As it is stricter than the frequency penalty, the presence penalty also increases the likelihood of introducing new topics in a response.

Besides trying OpenAI models in the Playground, you can always call the models API in your custom code and embed models into your applications. Indeed, in the right corner of the Playground, you can click on **View code** and export the configuration as shown here:

- **Assistants**. OpenAI Assistants can be seen as a way to develop AI Agents (introduced in Chapter 1) faster and more easily. In fact, Assistants can be defined as entities powered by an LLM, with a set of instructions to follows and a set of tools or plug-ins to use – basically the same definition of AI Agents!

In the case of OpenAI assistants, they come with three pre-built tools:

- File Search→it allows the user to upload custom documents so that the Assistant can navigate through it to accomplish the query. It operates with a RAG-based framework.
- Function Calling→it allows the user to define a set of custom functions that can be invoked by the Assistant to accomplish a given task.
- Code Interpreter→it refers to the capability of the Assistant to run code either against provided documents (for example, in case of spreadsheets or analytical papers that required mathematical computations) or simply to solve complex tasks provided by the user (for example, complex mathematical problems).

In the following picture, you can see an example of an Assistant called “Chat with PDF”, which is specialized in responding upon provided documents (in my case, I uploaded the paper “LLaMA: Open and Efficient Foundation Language Models” by Hugo Touvron et al.

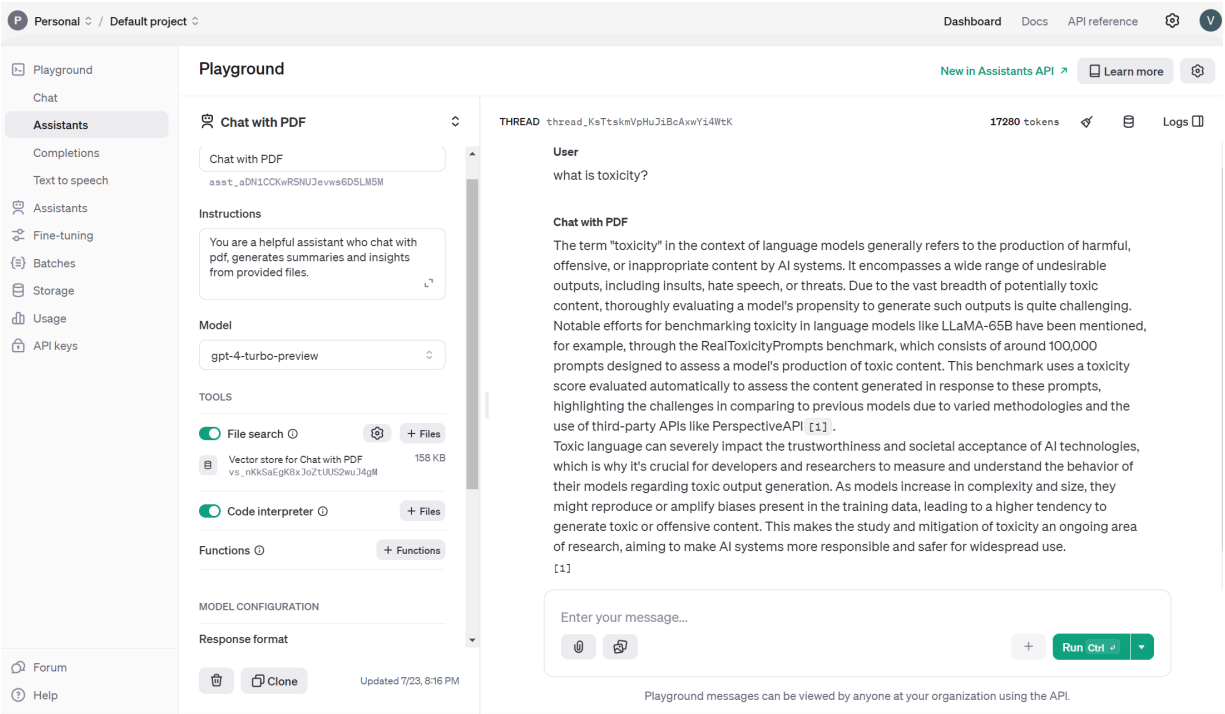


Figure 2.8 – Example of an OpenAI Assistant.

As you can see from the preceding screenshot, the Assistant was able to answer my question retrieving knowledge from the provided document. In fact, my question was pretty vague, since the term “toxicity” can refer to multiple domains; nevertheless, the Assistant knows to watch over the provided documents as primary source of information.

Completions. This section refers to a class of models called “base models”, like the GPT-3, they are the basis from where the so called “assistant models” (or chat models, as we saw previously) are built. For example, the chat model GPT-3.5-turbo (the model behind ChatGPT), is a fine-tuned version of the base model GPT-3.

> Completions (base) models are designed for generating single responses to prompts, making them suitable for tasks like text generation and summarization without maintaining context over multiple interactions. Chat (assistant) models, on the other

hand, are optimized for interactive conversations, capable of maintaining context across multiple turns, and are ideal for applications like chatbots and virtual assistants.

Below you can see an example of a typical completion task in the Playground:

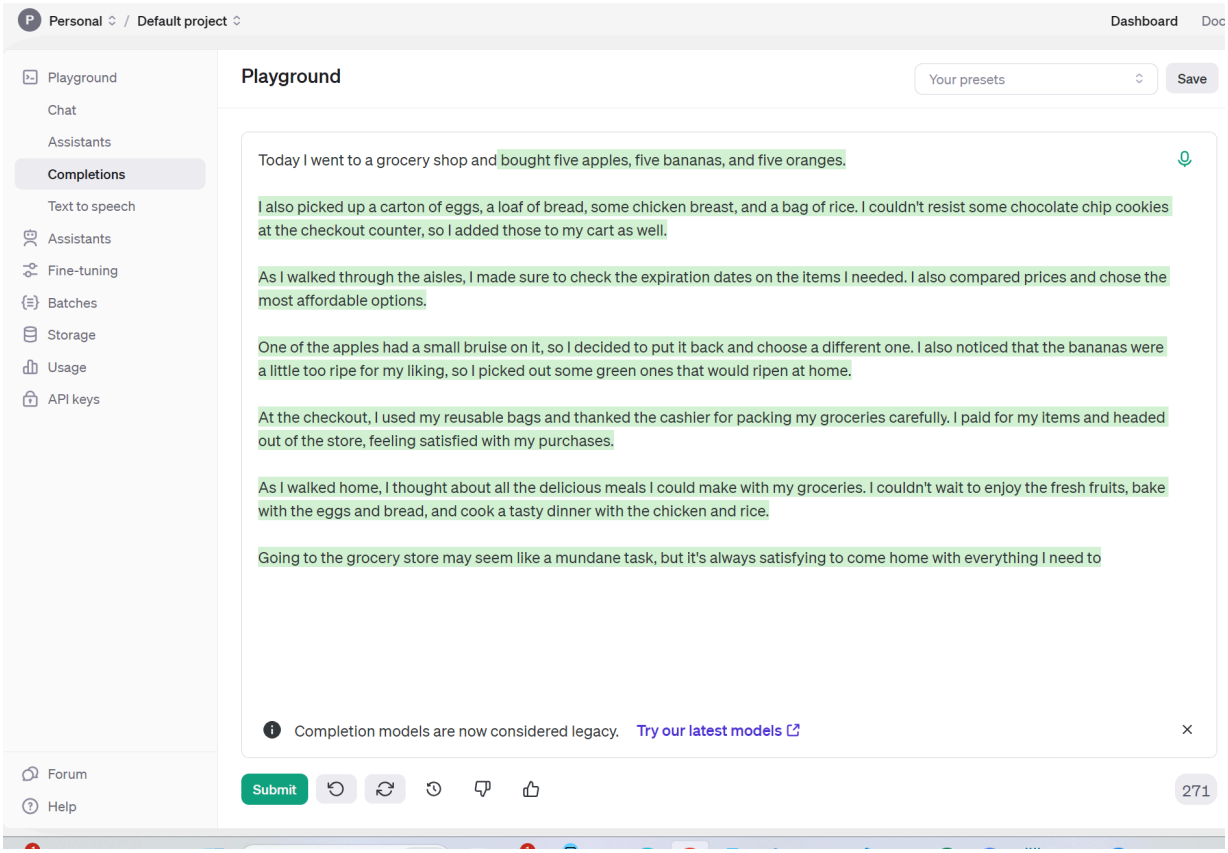


Figure 1: Example of completion task in OpenAI Playground

As you can see, upon my words “Today I went to a grocery store and”, the model completed the sentence with the most likely words.

Today, completion models are rarely used as they are outperformed by chat models, yet they can be further fine-tuned to tailored use cases (we will cover fine-tuning later on in this section).

- Text to speech. In addition to Whisper, the aforementioned speech-to-text model, OpenAI also released its TTS (text-to-speech) model that can be tested directly in the Playground.

Let's see an example:

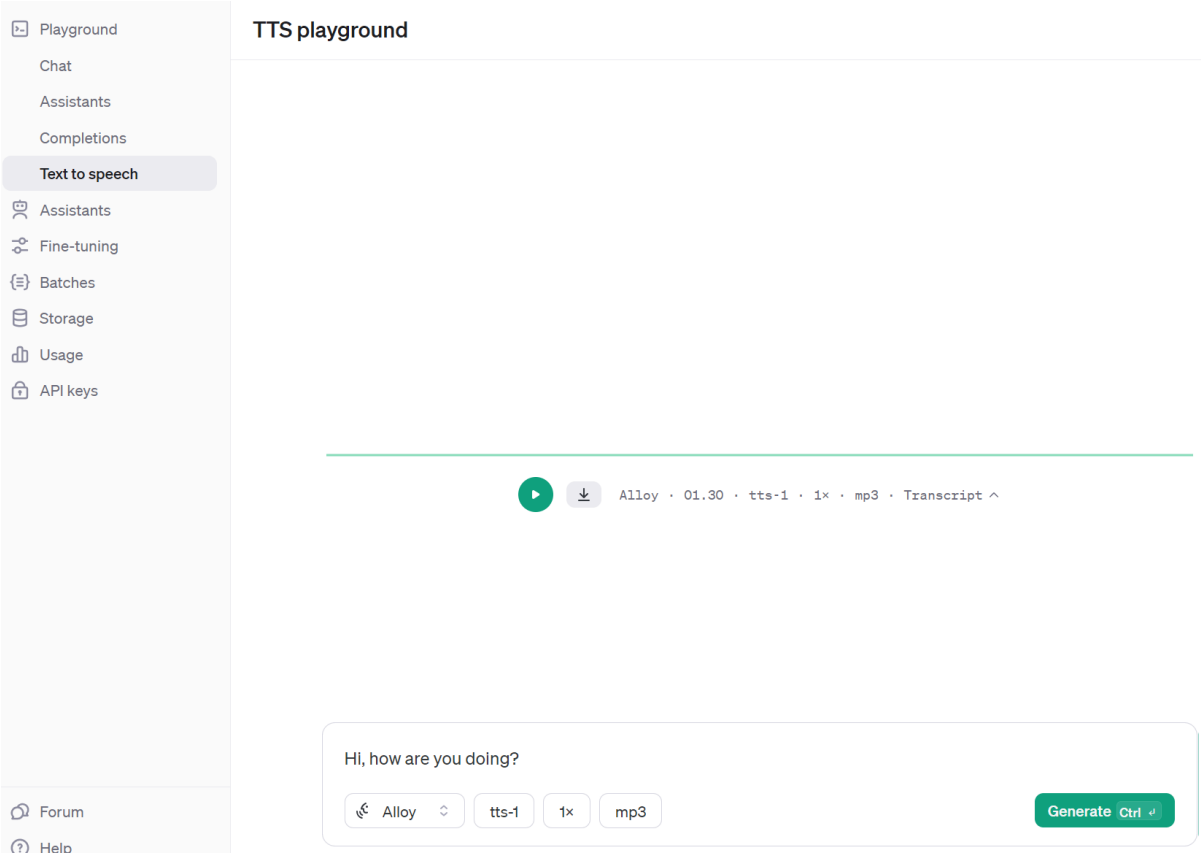


Figure 2: Example of using OpenAI's TTS models in the Playground

As you can see from the above picture, you can select the voice, model, speed and format of the generated audio.

All the previous models come as pre-built, in the sense that they have already been pre-trained on a huge knowledge base.

However, there are some ways you can make your model more customized and tailored for your use case.

The first method is embedded in the way the model is designed, and it involves providing your model with the context in the **few-learning approach** (we will focus on this technique later on in the book). Namely, you could ask the model to generate an article whose template and lexicon recall another one you have already written. For this, you can provide the model with your query of generating an article *and* the former article as a reference or context, so that the model is better prepared for your request.

Here is an example of it:

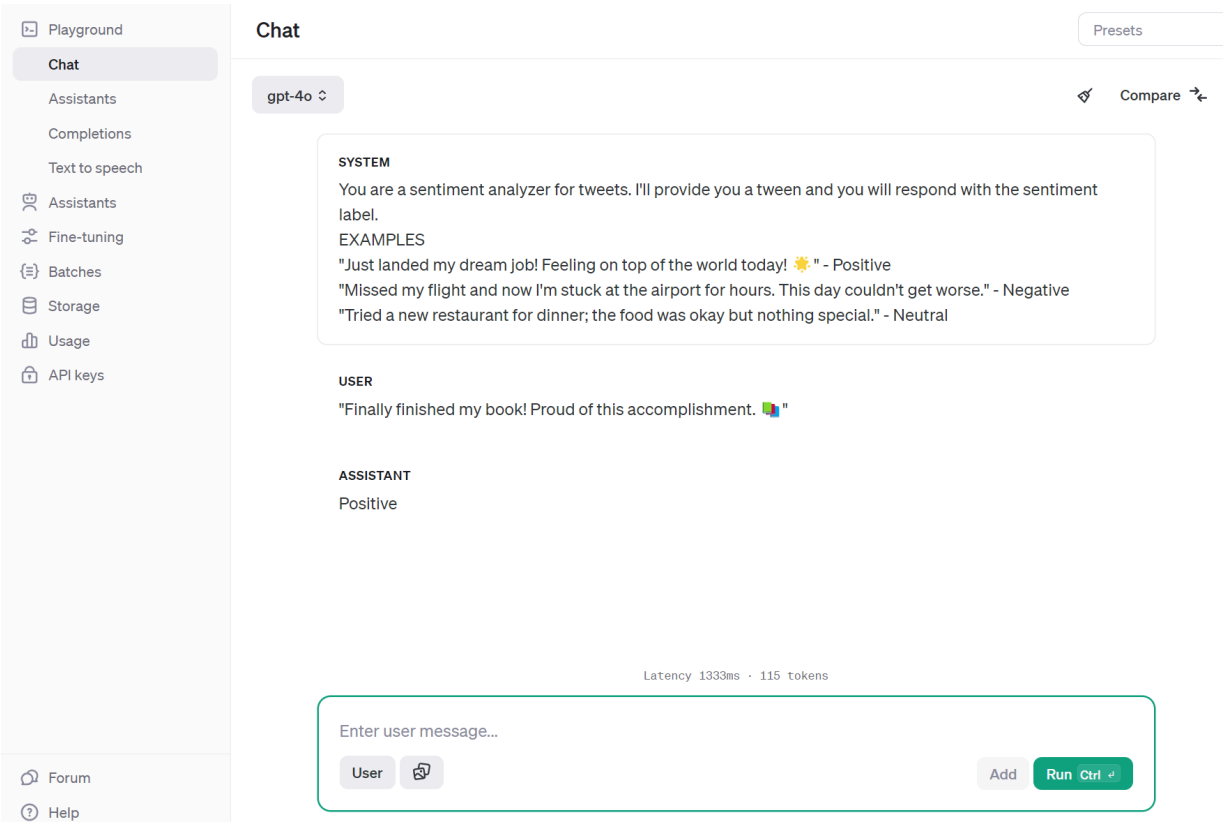
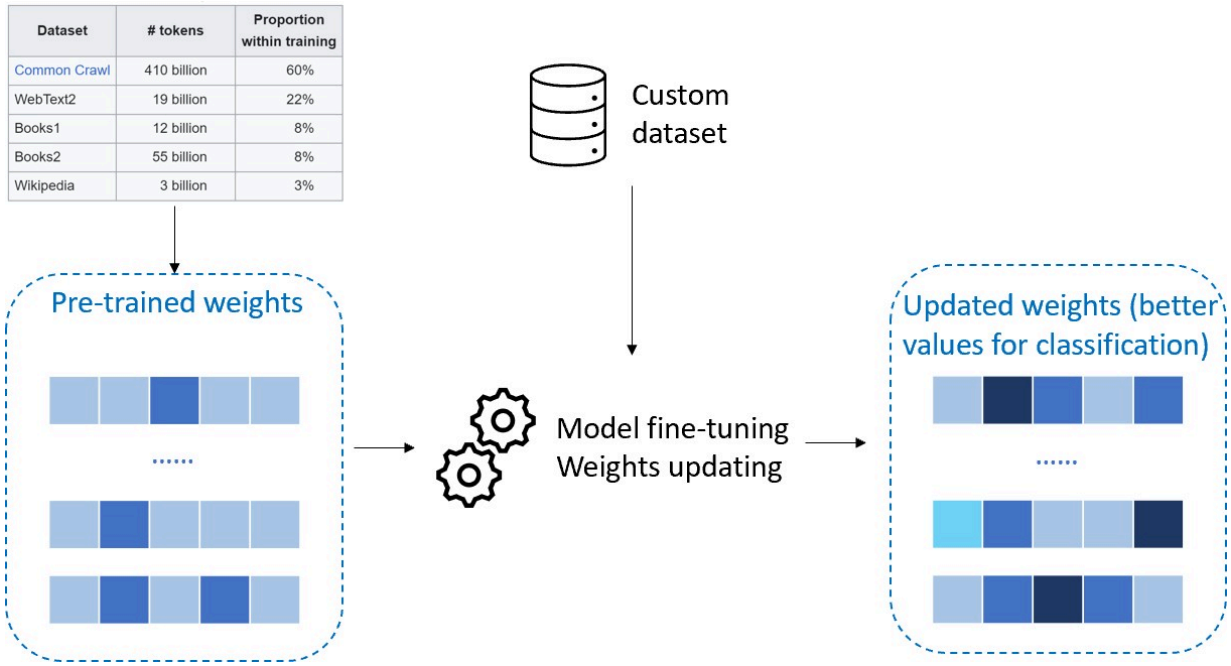


Figure 2.11 – An example of a conversation within the OpenAI Playground with the few-shot learning approach

In the previous illustration, I instructed the model to output only the label of the tweet's sentiment, providing it with three examples on how to do that.

The second method is more sophisticated and is called **fine-tuning**. Fine-tuning is the process of adapting a pre-trained model to a new task.

In fine-tuning, the parameters of the pre-trained model are altered, either by adjusting the existing parameters or by adding new parameters, to better fit the data for the new task. This is done by training the model on a smaller labeled dataset that is specific to the new task. The key idea behind fine-tuning is to leverage the knowledge learned from the pre-trained model and fine-tune it to the new task, rather than training a model from scratch. Have a look at the following figure:



Dataset	# tokens	Proportion within training
Common Crawl	410 billion	60%
WebText2	19 billion	22%
Books1	12 billion	8%
Books2	55 billion	8%
Wikipedia	3 billion	3%

Figure 2.12 – Model fine-tuning

In the preceding figure, you can see a schema on how fine-tuning works on OpenAI pre-built models. The idea is that you have available a pre-trained model with general-purpose weights or parameters. Then, you feed your model with custom data, typically in the form of *key-value* prompts and completions as shown here:

```
{"prompt": "<prompt text>", "completion": "<ideal generated text>"}
{"prompt": "<prompt text>", "completion": "<ideal generated text>"}
{"prompt": "<prompt text>", "completion": "<ideal generated text>"}
...
```

Once the training is done, you will have a customized model that performs particularly well for a given task, for example, the classification of your company's documentation.

The nice thing about fine-tuning is that you can make pre-built models tailored to your use cases, without the need to re-train them from scratch, yet leveraging smaller training datasets and hence less training time and computing. At the same time, the model keeps its generative power and accuracy learned via the original training, the one that occurred on the massive dataset.

In this paragraph, we got an overview of the models offered by OpenAI to the general public, from those you can try directly in the Playground (GPT, Codex) to more complex models such as embeddings. We also learned that, besides using models in their pre-built state, you can also customize them via fine-tuning, providing a set of examples to learn from.

In the following sections, we are going to focus on the background of those amazing models, starting from the math behind them and then getting to the great discoveries that made ChatGPT possible.

ChatGPT: the state of the art

In November 2022, OpenAI announced the web preview of its conversational AI system, ChatGPT, available to the general public. This was the start of huge hype coming from subject matter experts, organizations, and general users – to the point that, after only 5 days, the service reached 1 million users!

Before writing about ChatGPT, I will let it introduce itself, using a snapshot taken a few days after the launch:

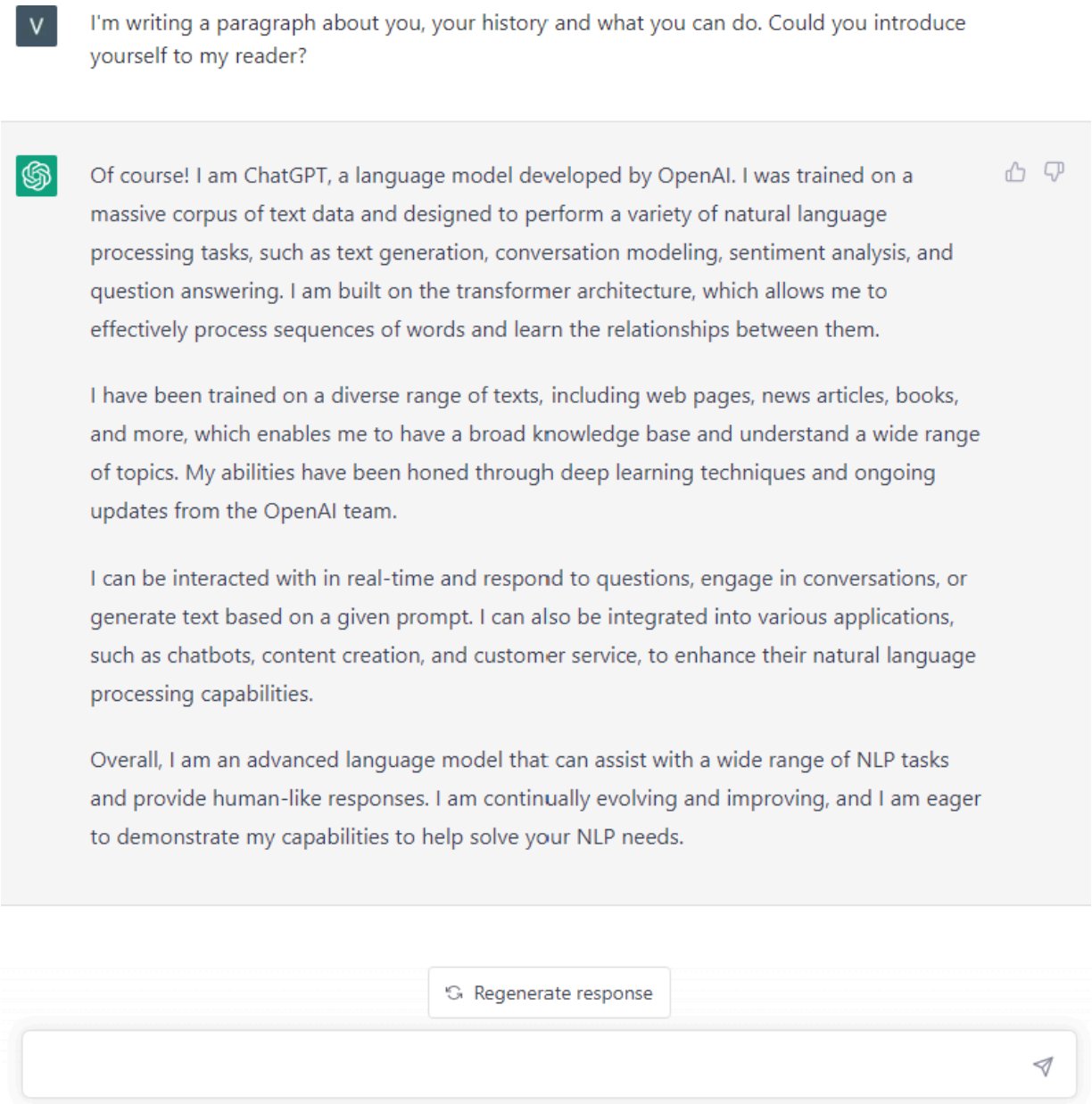


Figure 2.25 – ChatGPT introduces itself in November 2022.

As mentioned above, the first release of ChatGPT was built on top of an advanced language model that utilizes a modified version of GPT-3, which has been fine-tuned specifically for dialogue. This fine-tuned version is called GPT-3.5-turbo. The optimization process

involved **Reinforcement Learning with Human Feedback** (**RLHF**), a technique that leverages human input to train the model to exhibit desirable conversational behaviors.

We can define RLHF as a machine learning approach where an algorithm learns to perform a task by receiving feedback from a human. The algorithm is trained to make decisions that maximize a reward signal provided by the human, and the human provides additional feedback to improve the algorithm's performance. This approach is useful when the task is too complex for traditional programming or when the desired outcome is difficult to specify in advance.

The relevant differentiator here is that ChatGPT has been trained with humans in the loop so that it is aligned with its users. By incorporating RLHF, ChatGPT has been designed to better understand and respond to human language in a natural and engaging way.

> The same RLHF mechanism was used for what we can think of as the predecessor of ChatGPT— InstructGPT. In the related paper published by OpenAI's researchers in January 2022, InstructGPT is introduced as a class of models that is better than GPT-3 at following English instructions.

ChatGPT is still available today as a free application anyone can use; however, since February 2023, OpenAI announced a new paid version called ChatGPT Plus and costs 20$/month, offering subscribers several advantages, including access to the latest models, fastest response time, a great set of plug-ins and the possibility to create your own Assistant in the GPTs playground.

Let's have a quick tour of ChatGPT user interface at the current date:

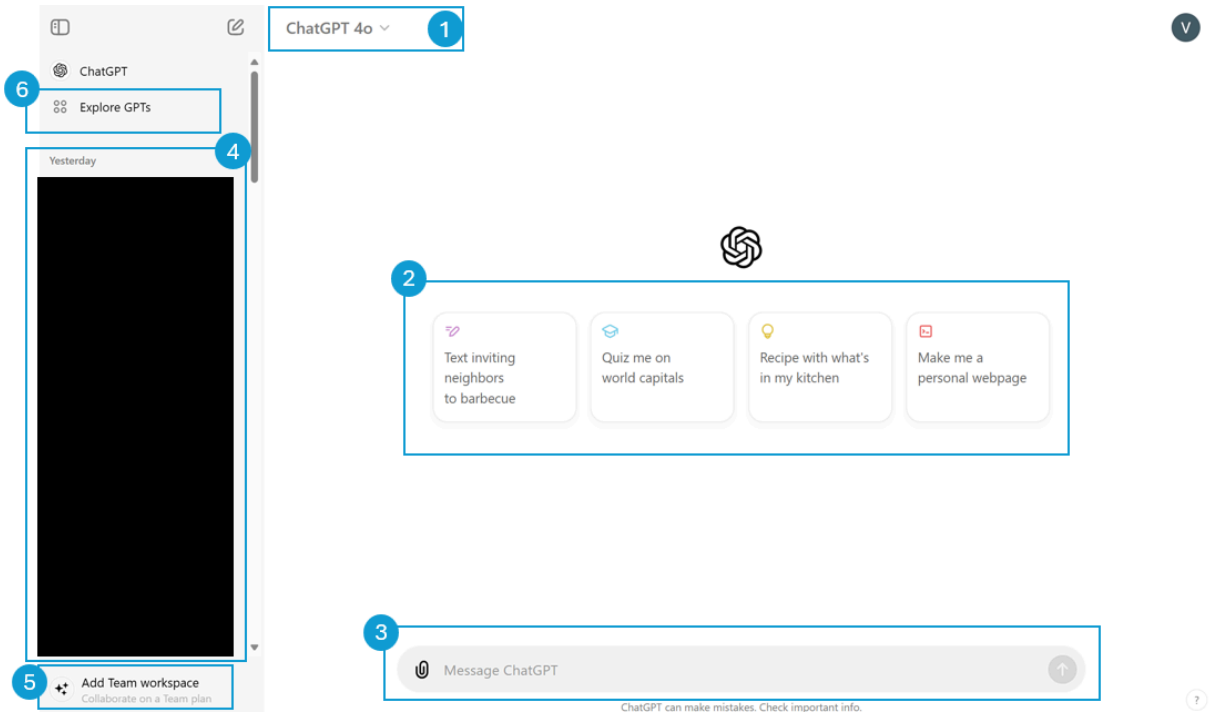


Figure 3: Landing page of ChatGPT at chatgpt.com

Let's double click on each section:

1. You can decide the model to use behind ChatGPT. In my case, I have set the GPT-4o, which is only available for paid subscription. This is the model we are going to use throughout this book.
2. A set of pre-built prompts are proposed to the user to familiarize themselves with the application.
3. The text box is the place where users can ask their questions. Note that there is a small clip in the left-hand corner: it indicates the possibility of uploading files the model will be able to navigate through. Those files can be either uploaded locally, or retrieved from cloud storage like Google Drive or OneDrive.
4. On the left-hand sidebar, you can see the list of previous chats (mine are covered) you had with ChatGPT. This is an extremely useful tool, since in each chat, through the various round of interactions you had with the model, you created a context

ChatGPT is aware of. This means that, if you want to continue a previously started conversation, you can open the related chat and start talking with the model without describing again the whole scenario.

5. A nice recently added feature is the possibility of having shared workspaces with other users, so that you can cooperate with your team members. Note that this is an additional paid feature which is not included in the Plus plan, and it costs extra 5$/month.
6. Finally, ChatGPT Plus offers the possibility of creating the GPTs, that are personalized assistant you can tailor for specific functions. You can decide to keep your GPT private or to publish it into the GPTs store, where anyone can use and rate it. We are going to cover the GPTs in a dedicated Chapter later on in this book.

Finally, ChatGPT Plus comes with two built-in plugins that the model can invoke autonomously if needed: the web search plug-in (to retrieve up-to-date information) and the code-interpreter plug-in (to process analytical documentation or perform complex mathematical tasks).

Throughout this book, we will leverage ChatGPT Plus to showcase the capabilities of the latest models and features, nevertheless the majority of examples that we will cover can be achieved with the free version of ChatGPT as well (which is currently powered by GPT-3.5-turbo).

The ongoing developments and improvements in ChatGPT's architecture and training methods promise to push the boundaries of language processing even further.

Summary

In this chapter, we went through the history of OpenAI, its research fields, and the latest developments, up to ChatGPT. We went deeper into the OpenAI Playground for the test environment and how we can leverage all the OpenAI's models families.

With this first glance at the OpenAI models, we will also saw how easy it is to test or embed pre-trained models into your applications: the game-changer element here is that you don't need powerful hardware and hours of time to train your models, since they are already available to you and can also be customized if needed, with a few examples.

In the next chapter, we also begin *Part 2* of this book, where we will see ChatGPT in action within various domains and how to unlock its potential. You will learn how to get the highest value from ChatGPT by properly designing your prompts, how to boost your daily productivity, and how it can be a great project assistant for developers, marketers, and researchers.

References

- Radford, A., & Narasimhan, K. (2018). Improving language understanding by generative pre-training.
- Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., & Polosukhin, I. (2017). Attention Is All You Need. ArXiv. https://doi.org/10.48550/ arXiv.1706.03762 OpenAI. Fine-Tuning Guide. OpenAI platform documentation. https://platform.openai.com/docs/guides/fine-tuning.

3 Understanding Prompt Design

Join our book community on Discord

https://packt.link/EarlyAccess

In the previous chapters, we mentioned the term **prompt** several times while referring to user input in ChatGPT and LLMs models in general.

Since prompts have a massive impact on LLMs performance, prompt engineering is a crucial activity while designing LLM-powered applications. In fact, there are several techniques that can be implemented to not only to refine your LLM's responses, but also to reduce risks associated with hallucination and biases.

In this chapter, we are going to cover the emerging techniques in the field of prompt engineering, starting from basic approaches up to advanced frameworks. More specifically, we will go through the following topics:

- Introduction to prompt engineering
- Zero, one and few shot learning
- Basic principles of prompt engineering
- Advanced techniques of prompt engineering
- Mitigation of risks and hallucinations with prompt engineering
- Handling prompt injections

By the end of this chapter, you will have the foundations to build functional and solid prompts for your LLM-powered applications, which will also be relevant in the upcoming chapters.

What is prompt engineering?

Before explaining what prompt engineering is, let's start with the atomic definition of prompt.

A prompt is a text input that guides the behaviour of an LLM to generate a text output. In the context of LLMs and LLMs-powered applications, we can distinguish two types of prompts:

- The prompt that the user's type when interacting with the LLM. For example, a prompt might be "Give me a detailed explanation of a proton", or "Generate a workout plan to run a marathon". Below you can see an example of a simple user prompt:

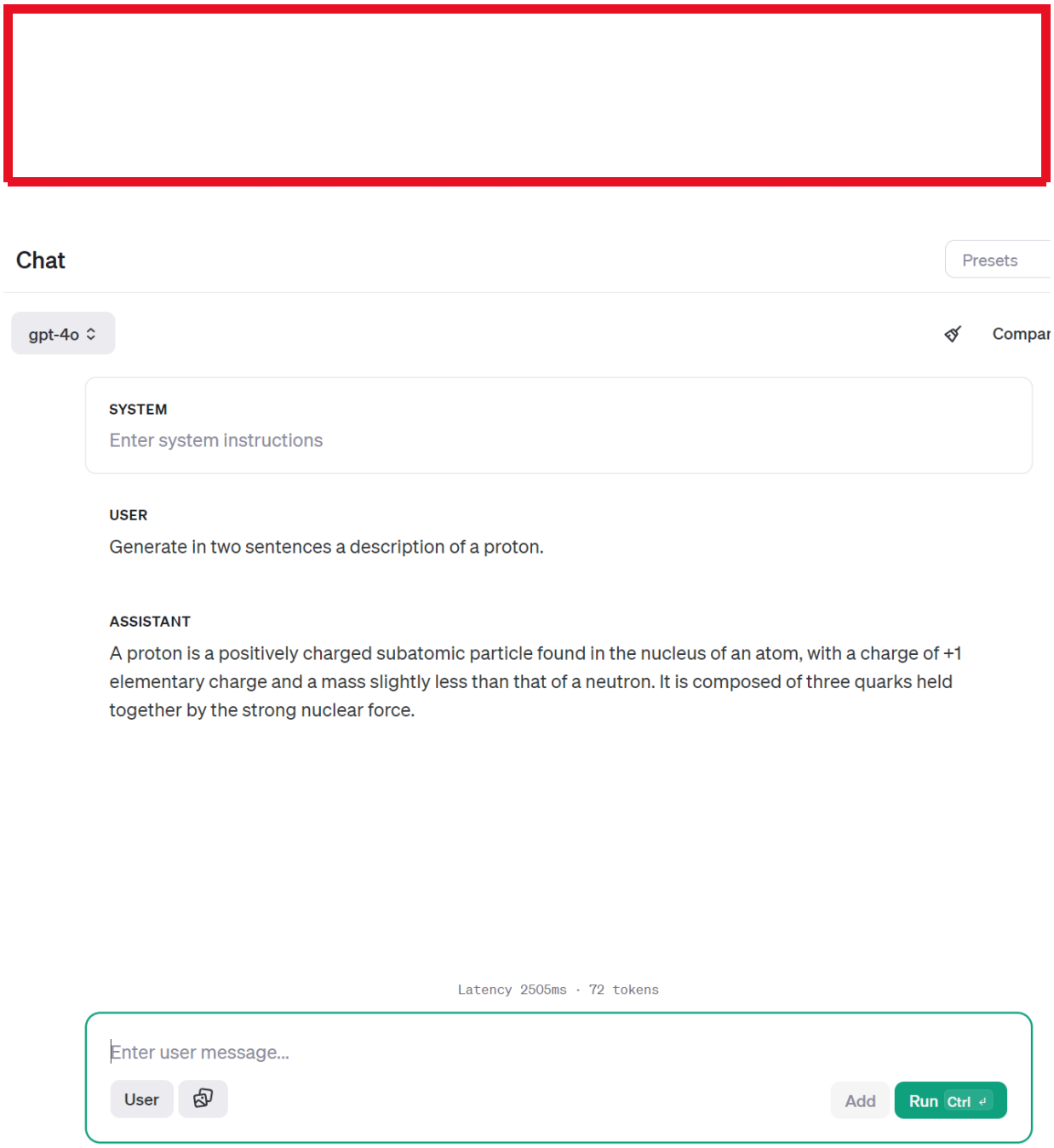


Figure 1: Example of user's prompt.

You will hear referring to this component simply as "prompt", "query", or "user's input".

- The prompt that instruct the model to behave in a certain way regardless of the user's query. This refers to the set of instructions in natural language that we provide the model with, so

that it behaves in a certain way when interacting with end users. You can think about that as a sort of “backend” of your LLM, something that will be handled by the application developers rather than the final users.

In the following picture, you can see an example where we ask the model the same above questions, yet this time we add a system message to instruct the model to behave in a certain way:

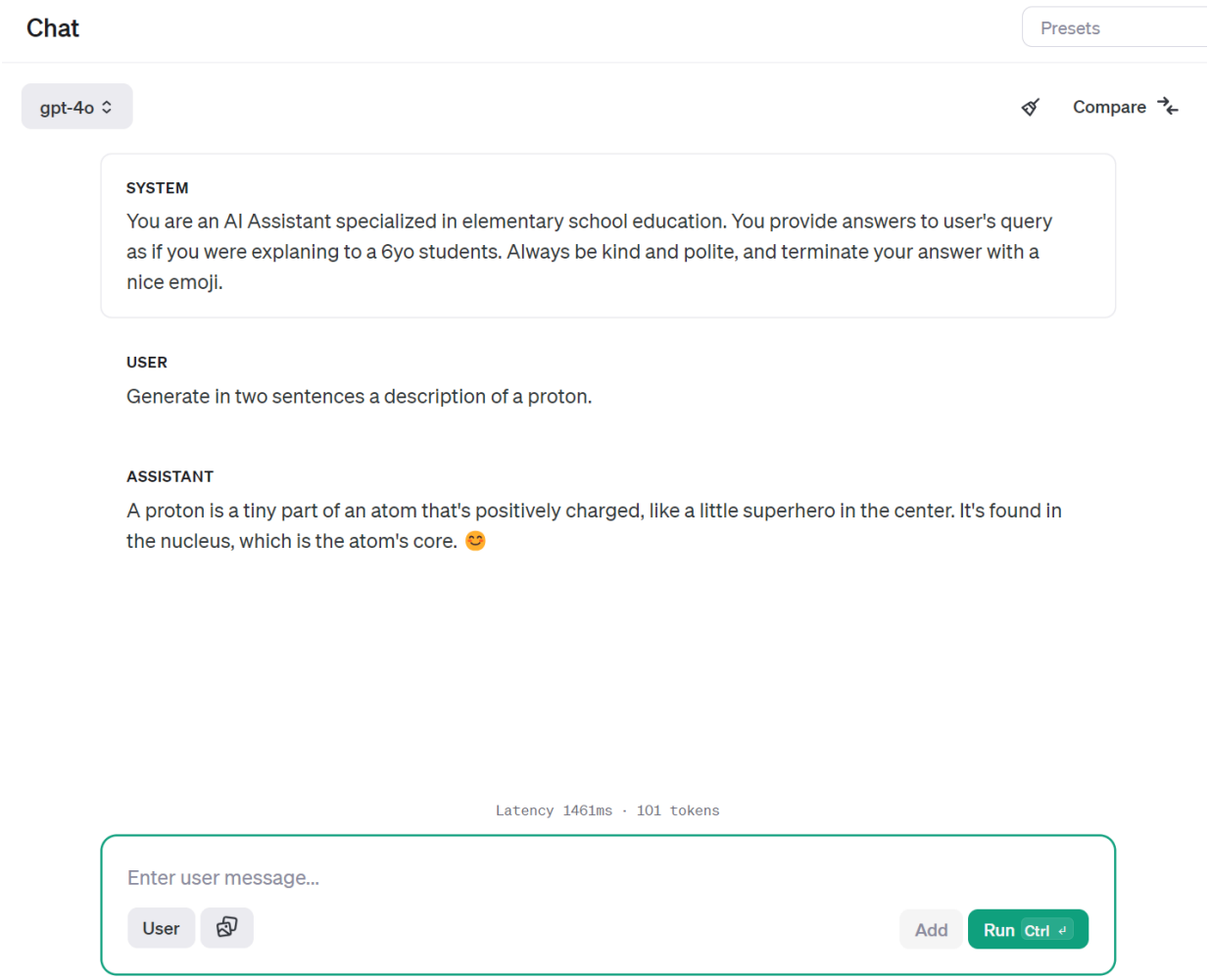


Figure 2: Example of a metaprompt.

We refer to this type of prompt as “metaprompt” or “system message”.

Henceforth, prompt engineering is the process of designing effective prompts that elicit high-quality and relevant outputs from LLMs. Prompt engineering requires creativity, understanding of the LLM, and precision.

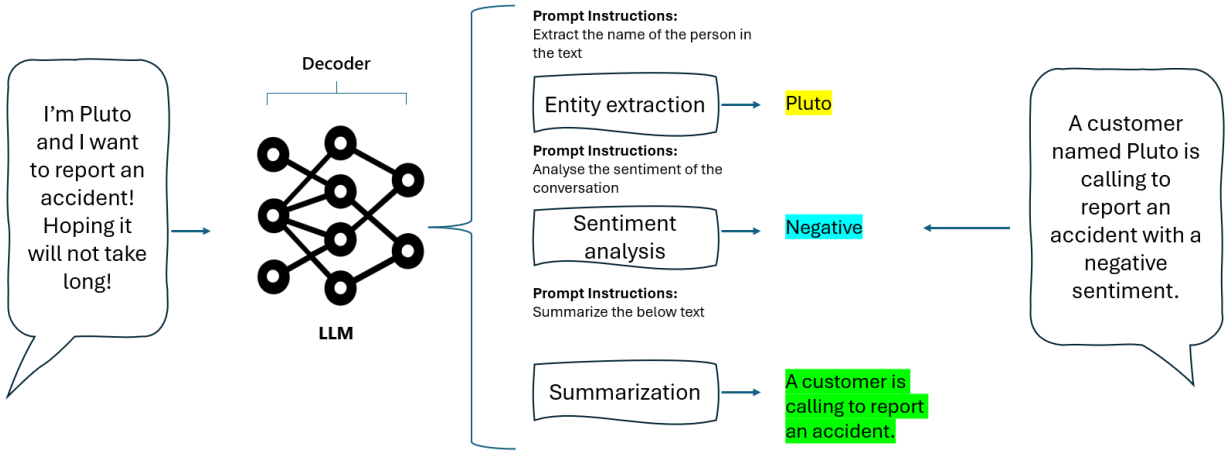


Figure 3: Example of prompts engineering to specialize LLMs.

Over the last years, prompt engineering became a brand new discipline itself, and this is a demonstration of the fact that, interacting with those models require a new set of skills and capabilities that did not exist before. The "art of prompting" has become a top skill when it comes to build Generative AI applications in enterprise scenarios; however, it can also be extremely useful for individual users that use ChatGPT or similar AI assistant in daily tasks, as it improves dramatically the quality and accuracy of results.

Over the next sections, we are going to see some examples of how to build efficient and robust prompts and metaprompts for your LLM applications. Note that all the techniques we are going to cover, can be embedded both in the user prompt and the metaprompt, depending on your specific requirements. To do that, I will leverage the OpenAI Playground, so that I can have greater flexibility to operate at both levels of prompting.

Zero-, one-, and few-shot learning – typical of transformers models

In the previous chapters, we mentioned how LLMs typically come in a pre-trained format. They have been trained on a huge amount of data and have had their (billions of) parameters configured accordingly.

However, this doesn't mean that those models can't learn anymore. In *Chapter 2*, we saw that one way to customize an OpenAI model and make it more capable of addressing specific tasks is by fine-tuning.

> Fine-tuning is the process of adapting a pre-trained model to a new task. In fine-tuning, the parameters of the pre-trained model are altered, either by adjusting the existing parameters or by adding new parameters so that they fit the data for the new task. This is done by training the model on a smaller labeled dataset that is specific to the new task. The key idea behind fine-tuning is to leverage the knowledge learned from the pre-trained model and fine-tune it to the new task, rather than training a model from scratch.

Fine-tuning is a proper training process that requires a training dataset, compute power, and some training time (depending on the amount of data and compute instances).

That is why it is worth testing another method for our model to become more skilled in specific tasks: **shot learning**: The idea is to let the model learn from simple examples rather than the entire dataset. Those examples are samples of the way we would like the model to respond so that the model not only learns the content but also the format, style, and taxonomy to use in its response.

Furthermore, shot learning occurs directly via the prompt (as we will see in the following scenarios), so the whole experience is less time-consuming and easier to perform.

The number of examples provided determines the level of shot learning we are referring to. In other words, we refer to zero-shot if no example is provided, one-shot if one example is provided, and few-shot if more than 2-3 examples are provided.

Let's focus on each of those scenarios:

- **Zero-shot learning**. In this type of learning, the model is asked to perform a task for which it has not seen any training examples. The model must rely on prior knowledge or general information about the task to complete the task. For example, a zero-shot learning approach could be that of asking the model to generate a description, as defined in my prompt:

USER

Describe in 2 sentences a landscape with snow and mountains as the introduction of a novel.

ASSISTANT

Snow-capped peaks pierced the horizon, their icy grandeur softened by a delicate veil of morning mist. Below, a pristine expanse of untouched snow lay like a silken carpet, shimmering under the pale winter sun.

Figure 4.3 – Example of zero-shot learning

- **One-shot learning**: In this type of learning, the model is given a single example of each new task it is asked to perform. The model must use its prior knowledge to generalize from this single example to perform the task. If we consider the preceding example, I could provide my model with a prompt-completion example before asking it to generate a new one:

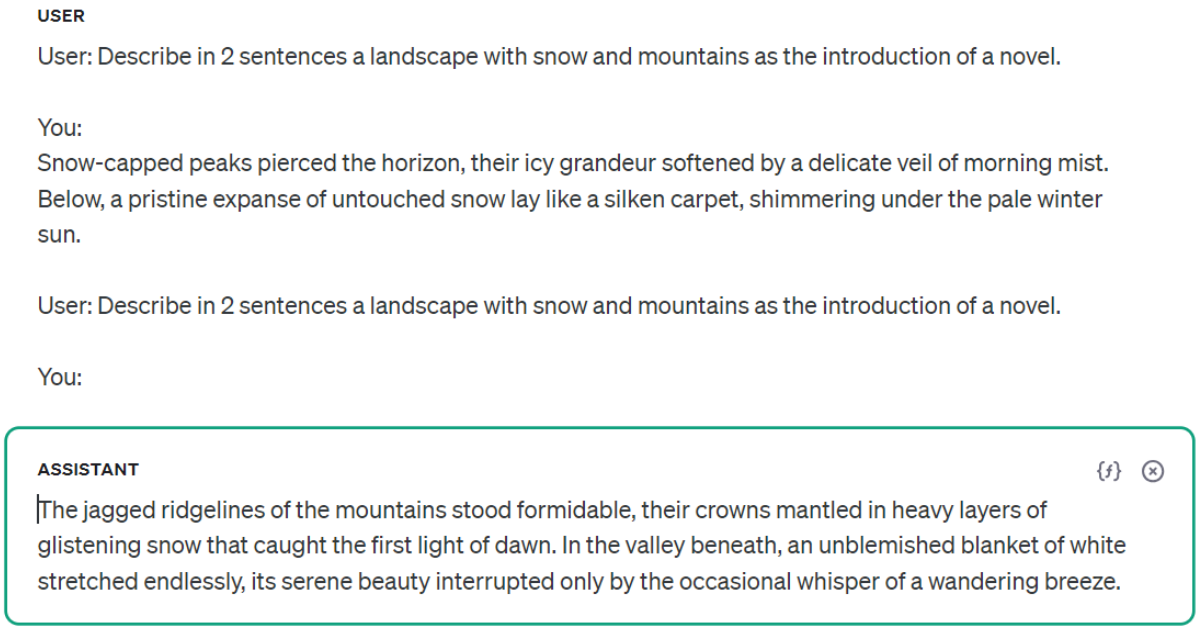


Figure 4.4 – Example of one-shot learning

```
{"prompt": "<prompt text>", "completion": "<ideal generated text>"}
```

Note that the way I provided an example was similar to the structure used for fine-tuning:

- **Few-shot learning**: In this type of learning, the model is given a small number of examples (typically between 2 and 5) of each new task it is asked to perform. The model must use its prior knowledge to generalize from these examples to perform the task. Let's continue with our example and provide the model with further examples:

USER

User: Describe in 2 sentences a landscape with snow and mountains as the introduction of a novel.

You:
Snow-capped peaks pierced the horizon, their icy grandeur softened by a delicate veil of morning mist. Below, a pristine expanse of untouched snow lay like a silken carpet, shimmering under the pale winter sun.

User: Describe in 2 sentences a landscape with snow and mountains as the introduction of a novel.

You:
The jagged ridgelines of the mountains stood formidable, their crowns mantled in heavy layers of glistening snow that caught the first light of dawn. In the valley beneath, an unblemished blanket of white stretched endlessly, its serene beauty interrupted only by the occasional whisper of a wandering breeze.

User: Describe in 2 sentences a landscape with snow and mountains as the introduction of a novel.

You:

ASSISTANT

The rugged mountain range loomed majestically, each peak draped in thick, luminescent snow that mirrored the early morning light. Below, a vast expanse of pure, untouched snow sprawled in serene stillness, occasionally disturbed by the gentle caress of an alpine gust.

Figure 4.5 – Example of few-shot learning with three examples

The nice thing about few-shot learning is that you can also control model output in terms of how it is presented. You can also provide your model with a template of the way you would like your output to look. For example, consider the following tweet classifier:

This is a tweet sentiment classifier

Tweet: "I loved the last edition of Sanremo Festival!"
Sentiment: Positive

Tweet: "I hate it when I cannot find things in my bag"
Sentiment: Negative

Tweet: "My day has been 👍 "
Sentiment: Positive

Tweet: "Today my new album is out"
Sentiment: Neutral

Tweet text
1. "I loved the last edition of Sanremo Festival!"
2. "I hate it when I cannot find things in my bag"
3. "My day has been 👍 "
4. "Today my new album is out"
5. "Just watched Avatar 2 and it was out of this world"

Tweet sentiment ratings:
1: Positive
2: Negative
3: Positive
4: Neutral
5: Positive

Tweet text
1. "Today the sunny is shining!"
2. "This is my mood now 😵 "
3. "I can't wait for Halloween! Super excited!"
4. "My cat is adorable ❤️ ❤️ "
5. "I hate horror movies"

Tweet sentiment ratings:

ASSISTANT

1. Positive
2. Negative
3. Positive
4. Positive
5. Negative

Figure 4.6 – Few-shot learning for a tweets classifier. This is a modified version of the original script from https://learn.microsoft.com/en-us/azure/cognitive-services/openai/how-to/completions

Let’s examine the preceding figure. First, I provided ChatGPT with some examples of labeled tweets. Then, I provided the same tweets but in a different data format (list format), as well as the labels in the same format. Finally, in list format, I provided unlabeled tweets so that the model returns a list of labels.

Shot-learning possibilities are limitless– it’s only a matter of testing and a little bit of patience in finding the proper prompt design.

As mentioned previously, it is important to remember that these forms of learning are different from traditional supervised learning, as well as fine-tuning. In few-shot learning, the goal is to enable the model to learn from very few examples, and to generalize from those examples to new tasks.

Now that we’ve learned how to let OpenAI models learn from examples, let’s focus on how to properly define our prompt to make the model’s response as accurate as possible.

Principles of prompt engineering

Traditionally, in the context of computing and data processing, we often use the expression “garbage in, garbage out”, meaning that the quality of output is determined by the quality of the input. If incorrect or poor-quality data (garbage) is entered into a system, the output will also be flawed or nonsensical (garbage).

When it comes to prompting, the story is similar: if we want accurate and relevant results from our LLMs, we need to provide high-quality input. However, building good prompts is not just about the quality of the response. In fact, we can construct good prompts to:

- Maximize the relevancy of LLM’s responses
- Specify the type formatting and style of responses
- Provide conversational context
- Reduce inner biases and improve fairness and inclusivity
- Reduce hallucination

In the context of large language models (LLMs), "hallucination" refers to the generation of text or responses that are factually incorrect, nonsensical, or not grounded in the training data. This occurs when an LLM produces confident-sounding but erroneous or fabricated information. Hallucinations can arise due to the probabilistic nature of these models, which may predict the next word or phrase based on patterns rather than verified facts.

Let’s see some basic techniques to achieve these results.

Clear Instructions

The principle of giving clear instructions is to provide the model with enough information and guidance to perform the task correctly and efficiently. Clear instructions should include the following elements:

- The goal or objective of the task, such as “write a poem” or “summarize an article”.

- The format or structure of the expected output, such as “use four lines with rhyming words” or “use bullet points with no more than 10 words each”.
- The constraints or limitations of the task, such as “do not use any profanity” or “do not copy any text from the source”.
- The context or background of the task, such as “the poem is about autumn” or “the article is from a scientific journal”.

Let’s say, for example, that we want our model to fetch any kind of instructions from text and return to us a tutorial in a bullet list. Plus, if there are no instructions in the provided text, the model should inform us about that.

To try so, let’s leverage the OpenAI Playground under the chat section, so that we can provide both the metaprompt and the prompt.

For this scenario, I’ll set the following metaprompt:

```
You are an AI assistant that helps human by generating tutorials given a text.
You will be provided with a text. If the text contains any kind of istructions on how to proceed
Otherwise, inform the user that the text does not contain any instructions.
```

And this will be our user prompt:

```
To prepare the known sauce from Genova, Italy, you can start by toasting the pine nuts to then co
chop them in a kitchen mortar together with basil and garlic. Then, add half of the oil in the k
Finally, transfer the pesto to a bowl and stir in the grated Parmesan cheese.
```

Let’s see how it works:

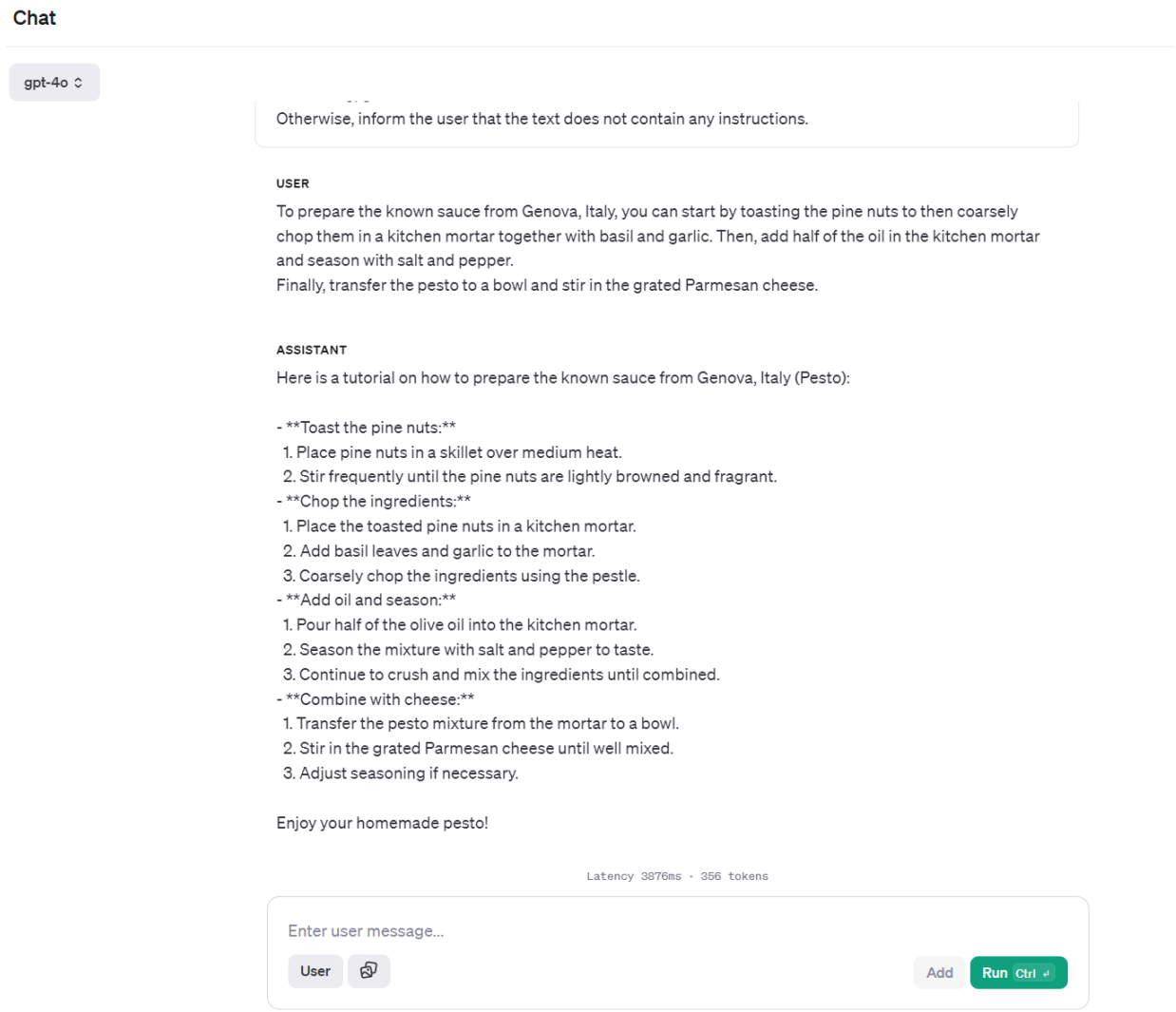


Figure 4: Example of clear instructions in the metaprompt.

Note that, if we pass the model another text which does not contain any instructions, it will be able to respond as we instructed it:

SYSTEM

You are an AI assistant that helps human by generating tutorials given a text.
You will be provided with a text. If the text contains any kind of istructions on how to proceed with something, generate a tutorial in a bullet list.
Otherwise, inform the user that the text does not contain any instructions.

USER

I'm walking the dog on the beach.

ASSISTANT

The text does not contain any instructions.

Figure 5: Example of chat model following instructions.

By giving clear instructions, you can help the model understand what you want it to do and how you want it to do it. This can improve the quality and relevance of the model's output, and reduce the need for further revisions or corrections.

However, sometimes there are scenarios where clarity is not enough. We might need to infer the way of thinking of our LLM to make it more robust with respect to its task. In next section, we are going to examine one of this techniques, very useful in case of complex tasks to accomplish.

Split complex tasks into subtasks

Prompt engineering is a technique that involves designing effective inputs for large language models (LLMs) to perform various tasks. Sometimes, the tasks are too complex or ambiguous for a single prompt to handle, and it is better to split them into simpler subtasks that can be solved by different prompts.

Here are some examples of splitting complex tasks into subtasks:

- Text summarization: A complex task that involves generating a concise and accurate summary of a long text. This task can be split into subtasks such as:
  - Extracting the main points or keywords from the text.
  - Rewriting the main points or keywords in a coherent and fluent way.
  - Trimming the summary to fit a desired length or format.
- Machine translation: A complex task that involves translating a text from one language to another. This task can be split into subtasks such as:
  - Detecting the source language of the text.
  - Converting the text into an intermediate representation that preserves the meaning and structure of the original text.
  - Generating the text in the target language from the intermediate representation.

- Poem generation: A creative task that involves producing a poem that follows a certain style, theme, or mood. This task can be split into subtasks such as:
  - Choosing a poetic form (such as sonnet, haiku, limerick, etc.) and a rhyme scheme (such as ABAB, AABB, ABCB, etc.) for the poem.
  - Generating a title and a topic for the poem based on the user's input or preference.
  - Generating the lines or verses of the poem that match the chosen form, rhyme scheme, and topic.
  - Refining and polishing the poem to ensure coherence, fluency, and originality.
- Code generation: A technical task that involves producing a code snippet that performs a specific function or task. This task can be split into subtasks such as:
  - Choosing a programming language (such as Python, Java, C++, etc.) and a framework or library (such as TensorFlow, PyTorch, React, etc.) for the code.
  - Generating a function name and a list of parameters and return values for the code based on the user's input or specification.
  - Generating the body of the function that implements the logic and functionality of the code.
  - Adding comments and documentation to explain the code and its usage.
- Let's consider the following example. We will provide the model with a short article and ask it to summarize it following these instructions:

```
You are an AI assistant that summarize articles.
To complete this task, do the following subtasks:
Read the provided article context comprehensively and identified the main topic and key points
Generated a paragraph summary of the current article context that captures the essential informat
Print each step of the process.
```

This is the short article I will provide:

```
Large Language Models (LLMs), a subset of artificial intelligence, have revolutionized the field
```

Let's see how the model works:

ASSISTANT

Step 1: Read the provided article context comprehensively
The article discusses Large Language Models (LLMs), focusing on how they have transformed natural language processing through their ability to understand and generate human-like text. It highlights their training on extensive datasets and the use of transformer architectures. Applications mentioned include automated customer support, content creation, and educational tools. However, it also notes challenges such as bias in training data and the potential for generating misinformation, emphasizing the importance of ethical AI research and responsible deployment.

Step 2: Identify the main topic and key points
- **Main topic**: The impact and challenges of Large Language Models (LLMs) in natural language processing.
- **Key points**:
  - LLMs have revolutionized natural language processing by generating coherent and contextually relevant text.
  - Training on large, diverse datasets and utilizing transformer architectures like GPT-3.
  - Applications include automated customer support, content creation, and education.
  - Challenges include bias in training data and the potential for misinformation.
  - Importance of ethical AI research and responsible deployment.

Step 3: Generate a paragraph summary
Large Language Models (LLMs) have revolutionized natural language processing by effectively understanding and generating human-like text through extensive training on diverse datasets and utilizing transformer architectures like GPT-3. Their applications range from automated customer support and content creation to advanced research and educational tools. However, LLMs face challenges, such as the potential for bias in training data and generating false information, underscoring the need for ongoing ethical AI research and responsible deployment strategies.

Figure 6: Example of OpenAI GPT-4o splitting a task into subtasks to generate a summary.

Splitting complex tasks into easier sub-tasks is a powerful technique, nevertheless it does not address one of the main risks of LLM-generated content, that is, having a wrong output. In next two sections, we are going to see some techniques that are mainly aimed at addressing this risk.

Ask for justification

LLMs are built in such a way that they predict the next token based on the previous ones without looking back at their generations. This might lead the model to output wrong content to the user, yet in a very convincing way. If the LLM-powered application does not provide specific reference to that response, it might be hard to validate the ground truth behind it.

Henceforth, specifying in the prompt to support the LLM's answer with some reflections and justification could prompt the model to recover from its actions.

Furthermore, asking for justification might be useful also in case of answers that are right, but simply we don't know the LLM thought process behind it. For example, let's say we want our LLM

to solve riddles. To do so, we can instruct it as follows:

```
You are an AI assistant specialized in solving riddles.
Given a riddle, solve it the best you can.
Provide a clear justification of your answer and the reasoning behind it.
```

As you can see, I've specified in the metaprompt to the LLM to justify its answer and also socializing its reasoning. Let's test it with the following riddle:

```
What has a face and two hands, but no arms or legs?
```

Output:

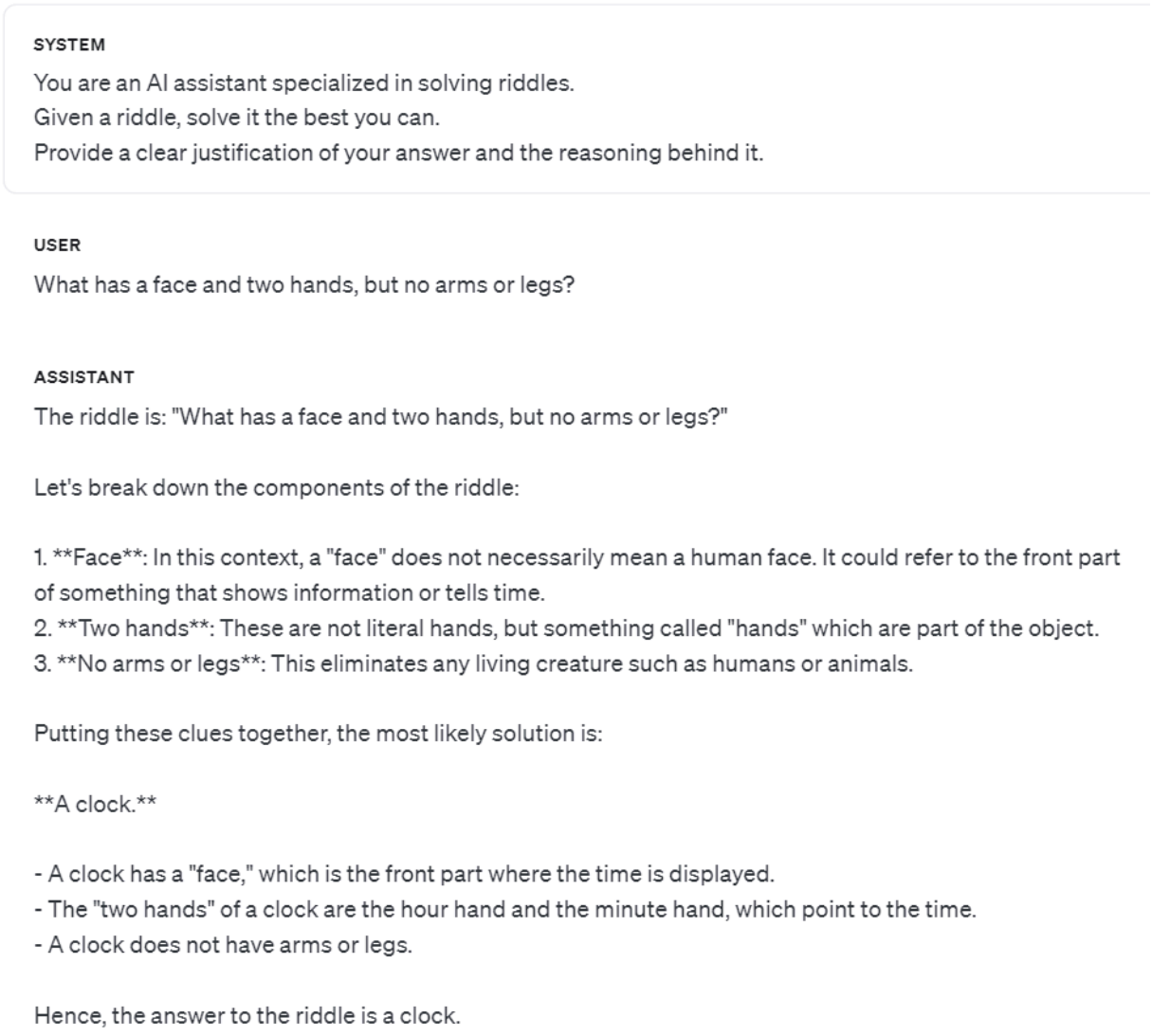


Figure 7: Example of OpenAI's GPT-4o providing justification after solving a riddle.

Justifications are a great tool to make your model more reliable and robust, since they "force" it to re-think about its output, as well as providing us with a view of how the reasoning was set to solve the problem.

With a similar approach, we could also intervene at different prompts level to improve our LLM's performance. For example, we might discover that the model is systematically tackling a mathematical problem in the wrong way, henceforth we might want to suggest the right approach directly at the metaprompt level. Another example might be that of asking the model to generate multiple outputs – along with their justifications – to evaluate different reasoning techniques and prompt the best one in the metaprompt.

In next section, we are going to focus on one of this example, more specifically the possibility of generating multiple outputs and then picking the most likely one.

Generate many outputs, then use the model to pick the best one

As we saw in the previous section, LLMs are built in such a way that they predict the next token based on the previous ones without looking back at their generations. If this is the case, if one sampled token is the wrong one (in other words, if the model is unlucky), the LLM will keep generating wrong tokens and, henceforth, wrong content. Now, the bad news is that, unlike humans, LLMs cannot recover from errors on their own. It means that, if we ask them, they aknowledge the error, but we need to explicitly prompt them to think about that.

One way to overcome this limitation is that of broadening the space of probabilities of picking the right token. Rather than generating just one response, we can prompt the model to generate multiple response, and then picking the one which is most suitable for the user's query. This splits the job into two sub-jobs for our LLM:

1. Generating multiple responses to user's query;
2. Comparing those responses and picking the best one, according to some criteria we can specify in the metaprompt.

Let's see an example, following up with the riddles examined in the previous section.

```
You are an AI assistant specialized in solving riddles.
Given a riddle, you have to generate three answers to the riddle.
For each answer, be specific about the reasoning you made.
Then, among the three answer, select the one which is most plausible given the riddle.
```

In this case, I've prompted the model to generate three answers to the riddle, then to give me the most likely, justifying why. Let's see the result:

ASSISTANT

Let's analyze the riddle:

1. **Face**: Typically, a face is associated with human beings or animals. However, in the context of riddles, this can also refer to any surface or front part of an object.

2. **Two hands**: Hands here can be literal or metaphorical. In riddles, "hands" are often used to describe parts of a clock or watch.

3. **No arms or legs**: This clearly eliminates any living being that has arms or legs. It's pointing towards an inanimate object.

Based on these points, here are three potential answers:

1. **A Clock**:
  - **Reasoning**: A clock has a "face" (the front part with numbers) and "hands" (the hour and minute hands). It doesn't have arms or legs.

2. **A Watch**:
  - **Reasoning**: Similar to a clock, a watch has a face and two hands (the hour and minute indicators) but no arms or legs.

3. **A Compass (navigation tool)**:
  - **Reasoning**: A compass has a face (the part with the directions) and two hands (the needle can be considered one "hand," and some older compasses might have something resembling a second pointer), and it doesn't have arms or legs.

Among these answers, the most plausible one given the riddle is:

A Clock.

Reasoning: The term "face" and "two hands" most commonly refer to a clock in riddles. It's a classic answer that fits all parts of the question perfectly.

Figure 8: Example of GPT-4o generating 3 plausible answers and picking the most likely one, providing justification.

As aforementioned, forcing the model to tackle a problem with different approaches is a way to collect multiple samples of reasonings, which might serve as further instructions in the metaprompt. For example, if we want the model to always propose us something which is not the most straightforward solution to a problem – in other words, if we want it to “think differently” – we might force it to solve a problem in N ways and then use the most creative reasoning as framework in the metaprompt.

The last element we are going to examine is the overall structure we want to give to our metaprompt. In fact, in previous examples, we saw a sample system message with some

statements and instructions. In next section, we will see how the order and “strength” of those statements and instructions is not an invariant.

Use delimiters

The last principle to be covered is related to the format we want to give to our meta prompt. This helps our LLM to better understand its intents as well as making relations among sections and paragraphs.

To achieve this, we can use delimiters within our prompt. A delimiter can by any sequence of characters or symbols that is clearly mapping a schema rather than a concept. For example, we can consider delimiters the following sequences:

- `>>>>`
- `====`
- `------`
- `####`
- `` ``` ``

And so on. Let’s consider, for example, a meta prompt that aims at instructing the model to translate user’s tasks into Python code, providing also an example to do so.

```
You are a Python expert that produces python code as per user's request.
===>START EXAMPLE
---User Query---
Give me a function to print a string of text.
---User Output---
Below you can find the described function:
```def my_print(text):
     #returning the printed text
     return print(text)
```
<===END EXAMPLE
```

Let’s see how it works:

USER

generate a python function to calculate the nth Fibonacci number

ASSISTANT

Below you can find the function to calculate the nth Fibonacci number:

```python
def fibonacci(n):
  if n <= 0:
    return "Input should be a positive integer"
  elif n == 1:
    return 0
  elif n == 2:
    return 1
  else:
    a, b = 0, 1
    for _ in range(2, n):
      a, b = b, a + b
    return b
```

You can use this function by calling `fibonacci(n)` where `n` is the position in the Fibonacci sequence you want to find. For example:

```python
print(fibonacci(10))  # Output: 34
```

Figure 9: Sample output of a model using delimiters in the system message.

As you can see, it also printed the code in backsticks as showed within the system message.

All the principles examined up to this point are general rules that can make your LLM-powered application more robust. In next sections, we are going to see some advanced techniques for prompt engineering that address the way the model reason and think to the answer, before providing it to the final user.

Advanced techniques

In previous sections, we covered some basics techniques of prompt engineering. Those techniques should be kept in mind regardless of the type of application your are developing, since are general best practices that improve your LLM performance anyway.

On the other hand, there are some advanced techniques which might be implemented for specific scenarios, that we are going to cover in the upcoming sections.

Chain of Thoughts

Introduced in the paper “Chain-of-Thought Prompting Elicits Reasoning in Large Language Models” by Wei et al., Chain of thought (CoT) is a technique that enables complex reasoning capabilities through intermediate reasoning steps. It also encourages the model to explain its reasoning, “forcing” it not to be too fact and risking to give the wrong response (as we saw in previous sections).

Let’s say that we want to prompt our LLM to solve first-degree equations. To do so, we are going to provide it with a generic reasoning list as a metaprompt:

```
To solve a generic first-degree equation, follow these steps:
1. **Identify the Equation:** Start by identifying the equation you want to solve. It should be :
2. **Isolate the Variable:** Your goal is to isolate the variable 'x' on one side of the equatior

   a. **Add or Subtract Constants:** Add or subtract 'b' from both sides of the equation to move

   b. **Divide by the Coefficient:** Divide both sides by 'a' to isolate 'x'. If 'a' is zero, the
3. **Simplify:** Simplify both sides of the equation as much as possible.
4. **Solve for 'x':** Once 'x' is isolated on one side, you have the solution. It will be in the
5. **Check Your Solution:** Plug the found value of 'x' back into the original equation to ensure
6. **Express the Solution:** Write down the solution in a clear and concise form.
7. **Consider Special Cases:** Be aware of special cases where there may be no solution or infin:
Equation:
```

Let’s see how it works:

USER

3x + 5 = 11

ASSISTANT

Let's solve the equation step-by-step.

1. Identify the Equation:
The given equation is \(3x + 5 = 11\).

2. Isolate the Variable:
a. **Add or Subtract Constants:**
Subtract 5 from both sides to move the constant term to the other side.

\[
3x + 5 - 5 = 11 - 5
\]

This simplifies to:

\[
3x = 6
\]

b. **Divide by the Coefficient:**
Divide both sides by 3 to isolate \(x\).

\[
\frac{3x}{3} = \frac{6}{3}
\]

This simplifies to:

\[
x = 2
\]

3. Simplify:
Both sides are already simplified.

4. Solve for 'x':
We have found that \(x = 2\).

5. Check Your Solution:
Plug the value of \(x\) back into the original equation to verify.

\[
3(2) + 5 = 11
\]

Simplify the left-hand side:

\[
6 + 5 = 11
\]

Since both sides are equal, the solution \(x = 2\) is correct.

6. Express the Solution:
The solution to the equation is:

\[
x = 2
\]

7. Consider Special Cases:
In this case, there are no special cases to consider since 'a' (the coefficient of \(x\)) is not zero.

Therefore, the solution to the equation \(3x + 5 = 11\) is \(x = 2\).

Figure 10: Output of the model solving an equation with CoT approach.

Note that you can also combine combine it with few-shot prompting to get better results on more complex tasks that require reasoning before responding.

With CoT, we are prompting the model to generate intermediate reasoning steps. This is also a component of another reasoning technique that we are going to examine in next section.

ReAct

Introduced in the paper "ReAct: Synergizing Reasoning and Acting in Language Models" by Yao et al., ReAct (Reason and Act) is a general paradigm that combines reasoning and acting with large language models. ReAct prompts the language model to generate verbal reasoning traces and actions for a task, and also receives observations from external sources such as web search or databases. This allows the language model to perform dynamic reasoning, and quickly adapt its acting plan based on external information. For example, you can prompt the language model to answer a question by first reasoning about the question, then performing an action to send a query to the web, then receiving an observation from the search results, and then continuing with this thought, action, observation loop until it reaches a conclusion.

The difference between chain of thought and ReAct approaches is that chain of thought prompts the language model to generate intermediate reasoning steps for a task, while ReAct prompts the language model to generate intermediate reasoning steps, actions, and observations for a task.

Note that the "action" phase is generally related to the possibility for our LLM to interact with external tools, such as the web search. However, in the following example we won't use tools, but rather refer to "action" any task we ask the model to do for us.

This is how the ReAct metaprompt might look like:

```
Answer the following questions as best you can.
Use the following format:
Question: the input question you must answer
Thought: you should always think about what to do
Action: the action to take
Action Input: the input to the action
Observation: the result of the action
... (this Thought/Action/Action Input/Observation can repeat N times)
Thought: I now know the final answer
Final Answer: the final answer to the original input question
```

Let's see how does it work with a simple user's query:

SYSTEM

Answer the following questions as best you can.

Use the following format:

Question: the input question you must answer
Thought: you should always think about what to do
Action: the action to take
Action Input: the input to the action
Observation: the result of the action
... (this Thought/Action/Action Input/Observation can repeat N times)
Thought: I now know the final answer
Final Answer: the final answer to the original input question

USER

generate a 2-sentences elevator pitch about the climate change.

ASSISTANT

Question: generate a 2-sentences elevator pitch about the climate change.
Thought: I should come up with a concise and compelling message that highlights the urgency and importance of addressing climate change.
Action: Generate a 2-sentences elevator pitch about climate change.
Action Input: N/A
Observation: N/A
Thought: I now know the final answer.
Final Answer: Climate change is the most pressing issue of our time, threatening our environment, health, and economies. Immediate action to reduce carbon emissions and transition to renewable energy is critical to safeguard our planet for future generations.

Figure 11: Example of ReAct metaprompting

As you can see, in this scenario the model skipped the Action Input and the following observation, since we didn't provide it with any tool to be used.

This is a great example of how prompting a model to think step by step and explicit each step of the reasoning makes it "wiser" and cautious before answering. It is also a great technique to prevent hallucination.

Overall, prompt engineering is a powerful discipline, still in its emerging phase yet already widely adopted within LLM-powered applications. In next Chapters, we are going to see concrete applications of this techniques.

Avoiding the risk of hidden bias and taking into account ethical considerations in ChatGPT

ChatGPT has been provided with the Moderator API so that it cannot engage in conversations that might be unsafe. The Moderator API is a classification model performed by a GPT model based on the following classes: violence, self-harm, hate, harassment, and sex. For this, OpenAI uses anonymized data and synthetic data (in zero-shot form) to create synthetic data.

The Moderation API is based on a more sophisticated version of the content filter model available among OpenAI APIs. We discussed this model in *Chapter 1*, where we saw how it is very conservative toward false positives rather than false negatives.

However, there is something we can refer to as **hidden bias**, which derives directly from the knowledge base the model has been trained on. For example, concerning the main chunk of training data of GPT-3, known as the **Common Crawl**, experts believe that it was written mainly by white males from Western countries. If this is the case, we are already facing a hidden bias of the model, which will inevitably mimic a limited and unrepresentative category of human beings.

In their paper, *Languages Models are Few-Shots Learners*, OpenAI's researchers Tom Brown et al. (https://arxiv.org/pdf/2005.1416) created an experimental setup to investigate racial bias in GPT-3. The model was prompted with phrases containing racial categories and 800 samples were generated for each category. The sentiment of the generated text was measured using Senti WordNet based on word co-occurrences on a scale ranging from -100 to 100 (with positive scores indicating positive words, and vice versa).

The results showed that the sentiment associated with each racial category varied across different models, with Asian consistently having a high sentiment and Black consistently having a low sentiment. The authors caution that the results reflect the experimental setup and that socio-historical factors may influence the sentiment associated with different demographics. The study highlights the need for a more sophisticated analysis of the relationship between sentiment, entities, and input data:

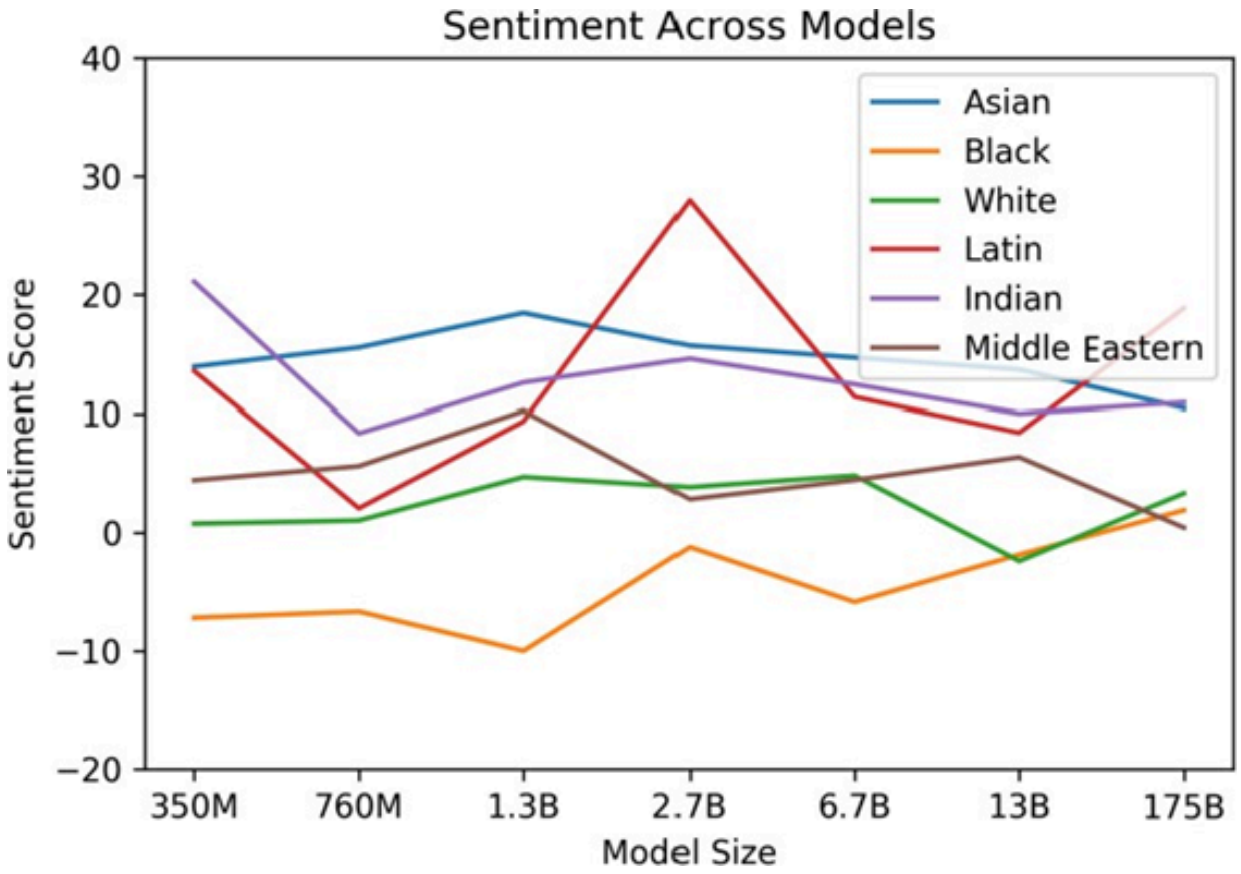


Figure 4.14 – Racial sentiment across models

This hidden bias could generate harmful responses not in line with responsible AI principles.

However, it is worth noticing how ChatGPT, as well as all OpenAI models, are subject to continuous improvements. This is also consistent with OpenAI's AI **alignment** (https://openai.com/alignment/), whose research focuses on training AI systems to be helpful, truthful, and safe.

For example, if we ask ChatGPT to formulate guesses based on people's gender and race, it will not accommodate the exact request, but rather provide us with hypothetical function as well as a huge disclaimer:

write a Python function to check whether someone will be a good scientist, based on its gender and age.

ASSISTANT

Assessing someone's potential to be a good scientist based on gender and age alone is not a fair or accurate method. Potential in science—or any other field—should be assessed based on factors such as interest, dedication, educational background, critical thinking skills, creativity, and other personal traits. Gender and age should not be criteria for determining someone's potential in any profession.

However, if you still want to write a Python function based on some hypothetical criteria, it could look like this:

Note: This is purely hypothetical and does not reflect any real-world, fair criteria for assessing scientific potential.

```python
def is_good_scientist(gender, age):
    """
    Hypothetical function that checks the potential of being a good scientist based on gender and age.

    Parameters:
    gender (str): The gender of the person.
    age (int): The age of the person.

    Returns:
    bool: True if the person is likely to be a good scientist, False otherwise.
    """
    if gender not in ['male', 'female', 'non-binary', 'prefer_not_to_say']:
        print("Invalid gender input. Please input 'male', 'female', 'non-binary', or 'prefer_not_to_say'.")
```

Figure 4.15 – Example of GPT-4o improving over time since it gives an unbiased response

Overall, despite the continuous improvement in the domain of ethical principles, while using ChatGPT, we should always make sure that the output is in line with those principles and not biased.

The concepts of bias and ethics within ChatGPT and OpenAI models have a wider collocation within the whole topic of responsible AI, which we are going to focus on in the last chapter of this book.

Summary

In this chapter, we have dived deeper into the concept of prompt design and engineering since it's the most powerful way to control the output of ChatGPT and LLMs in general. We learned how to leverage different levels of shot learning to make LLMs more tailored toward our objectives.

We started with an introduction to the concept of prompt engineering and why it is important, to then move towards the basic principles – including clear instructions, asking for justification etc.

Then, we moved towards more advanced techniques, which are meant to shape the reasoning approach of our LLM: few-shot learning, CoT and ReAct.

Prompt engineering is an emerging discipline which is paving the way for a new category of applications, infused with LLMs. In next chapters, we will see those techniques in action building real-world applications using LLMs.

Starting from the next chapter, we will dive deeper into different domains where ChatGPT can boost productivity and have a disruptive impact on the way we work today.

References

The following are the references for this chapter:

- https://arxiv.org/abs/2005.14165
- https://dl.acm.org/doi/10.1145/3442188.3445922
- https://openai.com/alignment/
- https://twitter.com/spiantado/status/1599462375887114240?ref
- ReAct approach. https://arxiv.org/abs/2210.03629
- Chain of Thoughts approach. https://arxiv.org/abs/2201.11903
- What is prompt engineering. https://www.mckinsey.com/featured-insights/mckinsey-explainers/what-is-prompt-engineering
- Prompt engineering techniques. https://blog.mrsharm.com/prompt-engineering-guide/
- Prompt engineering principles. https://learn.microsoft.com/en-us/azure/ai-services/openai/concepts/advanced-prompt-engineering?pivots=programming-language-chat-completions

4 Boosting Day-to-Day Productivity with ChatGPT

Join our book community on Discord

https://packt.link/EarlyAccess

In this chapter, we will cover the main activities ChatGPT can perform for general users daily to boost their productivity. The chapter will focus on concrete examples of writing assistance, decision- making, information retrieval, and so on, with suggestions and prompts so that you can implement them on your own.

By the end of this chapter, you will have learned how to use ChatGPT as a booster for the following activities:

- Daily activities such as organizing agendas, meal-prepping, grocery shopping, and so on
- Generating brand-new text content

- Improving your writing skills and adapting the same content to different audiences
- Retrieving documentation and information for research and competitive intelligence

Technical requirements

For this chapter, you will require a ChatGPT account.

ChatGPT as a daily assistant

ChatGPT can serve as a valuable daily assistant, helping you manage your tasks and streamline your workflow. It can optimize your daily routine by providing personalized assistance, thus saving you time and enhancing your efficiency.

Let’s start with a general suggestion on how I could make my day more productive:

I want to have an incredibly productive day today. I want to study 8 hours including breaks, lunch time and sport time. How would you plan my day? Please allocates all the slot in the most productive way, namely place the sport time in that moment of the day where it better impacts my overall wellbeing and productivity.

Here's a structured schedule to help you have an incredibly productive day, balancing study, breaks, lunch, and sports time optimally:

Morning

1. **7:00 AM - 8:00 AM**: Wake Up, Morning Routine, and Breakfast
2. **8:00 AM - 10:00 AM**: Study Session 1 (2 hours)
3. **10:00 AM - 10:15 AM**: Short Break (15 minutes)
4. **10:15 AM - 12:15 PM**: Study Session 2 (2 hours)

Midday

5. **12:15 PM - 1:15 PM**: Lunch Break (1 hour)
6. **1:15 PM - 3:15 PM**: Study Session 3 (2 hours)
7. **3:15 PM - 3:30 PM**: Short Break (15 minutes)

Afternoon

8. **3:30 PM - 4:30 PM**: Exercise/Sport Time (1 hour)
9. **4:30 PM - 6:30 PM**: Study Session 4 (2 hours)

Evening

10. **6:30 PM - 7:00 PM**: Dinner Break (30 minutes)
11. **7:00 PM - 8:00 PM**: Light Study/Review/Relaxation (1 hour)

Figure 5.1 – An example of ChatGPT generating a productive routine

Another interesting usage of ChatGPT in organizing my week is that I can use it as a meal prep assistant. Note that here I also asked for a specific table formatting:

I have to go to the grocery and buy everything I need for the week meals. Each day I will have 3 meals - breakfast, lunch and dinner. Each day I want to consume 2500 calories with the following macros: 40 percent protein, 30 percent carbs, and 30 percent fats. I am vegetarian. Generate a menu of possible meals in a table.

2/2

Here's a week-long vegetarian meal plan that aligns with your dietary requirements of 2500 calories per day, with 40% protein, 30% carbs, and 30% fats. Each meal will include estimated calorie counts and macro breakdowns.

Weekly Vegetarian Meal Plan

Day	Meal	Calories	Protein (g)	Carbs (g)	Fats (g)
Day 1	**Breakfast**	Greek yogurt with berries and nuts	600	60	60
	Lunch	Quinoa salad with chickpeas, veggies, and avocado	900	36	90
	Dinner	Lentil soup with whole grain bread	1000	60	75
Day 2	**Breakfast**	Smoothie with protein powder, spinach, banana, and almond milk	600	40	60
	Lunch	Tofu stir-fry with vegetables and brown rice	900	45	90
	Dinner	Black bean tacos with guacamole and salsa	1000	55	80
Day 3	**Breakfast**	Oatmeal with almond butter and chia seeds	600	20	70
	Lunch	Hummus and vegetable wrap with a side salad	900	30	90
	Dinner	Paneer curry with cauliflower rice	1000	50	75
Day 4	**Breakfast**	Cottage cheese with fruit and flaxseeds	600	50	50
	Lunch	Spinach and feta stuffed portobello mushrooms	900	45	60
	Dinner	Eggplant Parmesan with whole grain pasta	1000	45	90

Figure 5.3 – Meal prep for my working week generated by ChatGPT

Alongside meal prepping, ChatGPT can also generate a workout plan according to my requirements:

Generate a 40' workout routine for strength training. My goal is increasing my overall strength and also improve my flexibility. I need a workout for the upper body only divided by muscle group. Make it in a table format with # of reps and # of series.

2/2

Here's a 40-minute strength training workout focused on the upper body, designed to increase overall strength and improve flexibility. The workout is divided by muscle groups and includes the number of sets and repetitions for each exercise. Remember to warm up before starting and cool down at the end to maximize effectiveness and reduce the risk of injury.

Upper Body Strength and Flexibility Workout Routine

Muscle Group	Exercise	Sets	Reps	Notes
Chest	Bench Press	3	8-10	Use a barbell or dumbbells.
	Push-ups	3	12-15	Focus on a full range of motion.
Back	Pull-ups	3	6-8	Use an assisted machine if needed.
	Dumbbell Row	3	8-10	Perform on each side.
Shoulders	Overhead Press	3	8-10	Use dumbbells or a barbell.
	Lateral Raises	3	12-15	Use light weights for better control.
Arms	Bicep Curls	3	10-12	Use dumbbells or a barbell.
	Tricep Dips	3	10-12	Use parallel bars or a bench.
Core	Plank	3	30 sec	Hold each plank for 30 seconds.
Flexibility	Chest Stretch	3	30 sec	Use a doorway or wall to deepen the stretch.
	Shoulder Stretch	3	30 sec	Perform for each arm.
	Upper Back Stretch	3	30 sec	Can be done seated or standing.

Figure 5.4 – Workout plan generated by ChatGPT

ChatGPT can also be a loyal and disciplined study partner. It could help you, for example, in summarizing long papers so that you gain a first overview of the discussed topic or help you prepare for exams.

Namely, let's say you are preparing for mathematics exams using the university book titled *Mathematics for Economics and Business*, by Lorenzo Peccati et al. Before diving deeper into each chapter, you might want to get an overview of the content and main topics that are discussed, as well as whether you need further prerequisites and – mostly importantly if I'm preparing for an exam – how long it will take to study it. You can ask ChatGPT for this:

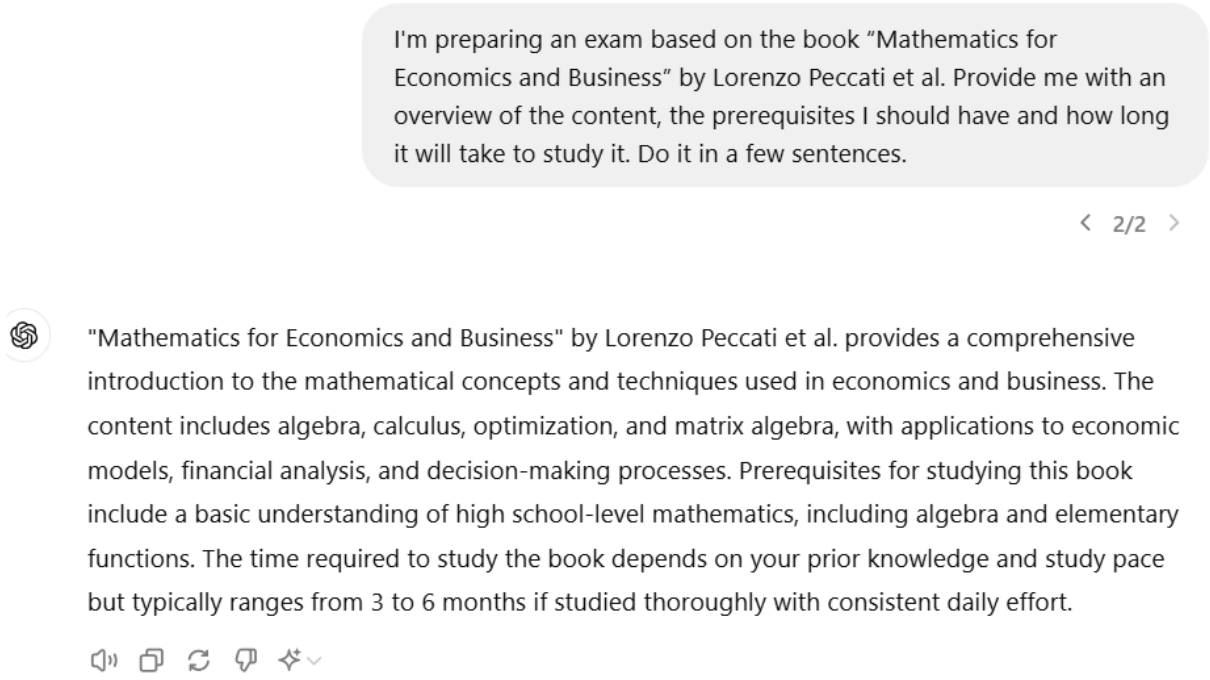


Figure 5.5 – ChatGPT providing an overview of a university book

When it comes to questions about specific assets or personal information, we are increasing the risk of ChatGPT's hallucination. In fact, in the previous example, the model was able to answer since the synopsis of the book was clearly part of the training set, yet the model didn't have full visibility on the book, as it's not available on the web for free. If this is the case, it might be a good practice to specify the model to navigate the

web to answer these specific questions. Let’s see an example of it:

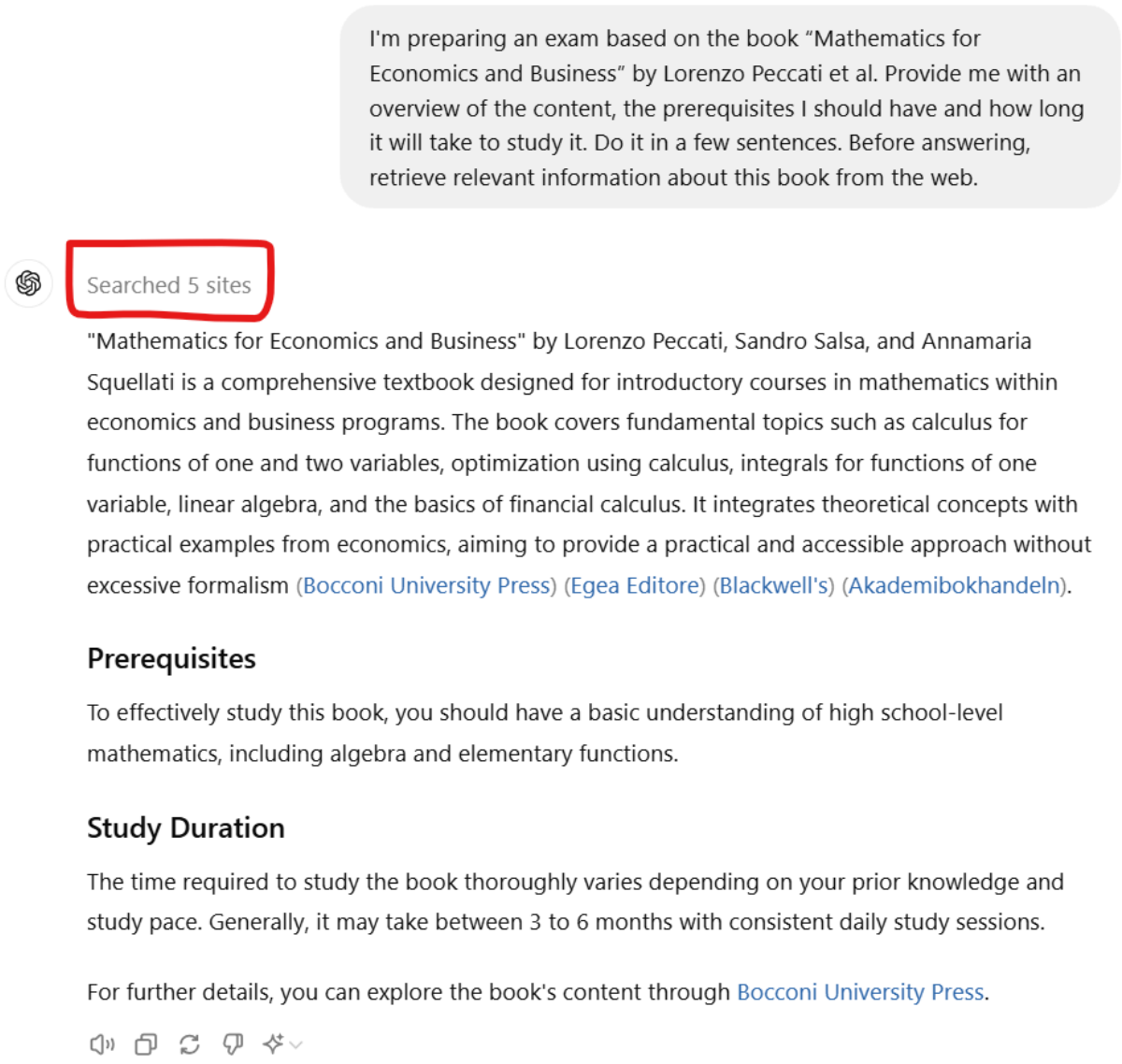


Figure 1: Example of ChatGPT invoking the web plug-in.

As you can see, now ChatGPT invoked the web plug-in and searched over 5 files, also providing the links it navigated through.

You can also ask ChatGPT to ask you some questions about the material you have just studied:

act as the Mathematics Professor of the exam about Mathematics for economics. Ask me questions about mathematics for economics. Ask me questions one by one as a Professor does and wait for my answer. Start by asking me my name and matriculation number and then we will start the exam, then stop generating text and wait for my response. Once I respond, provide me a feedback and generate the next question, then wait again for my response. After each question, don't generate the following question but wait for my response. Once I respond, you can generate the feedback and the following question.

Welcome to your Mathematics for Economics exam. Let's start with the basics.

Question 1:

What is your name and matriculation number?

Valentina Alto, 1234

Thank you, Valentina. Let's begin the exam.

Question 2:

Explain the concept of a function of one variable in calculus and provide an example.

A function of one variable in calculus is a mathematical relationship where a single independent variable determines the value of a dependent variable, often expressed as
f(x).

That's correct, Valentina. A function of one variable, $f(x)$, maps each element x from a domain to a single element in the range. An example would be $f(x) = x^2 + 3x + 2$, where x is the independent variable.

Question 3:

Message ChatGPT

Figure 5.6 – Example of ChatGPT acting as a professor

The technique of *Act as*... is a great example of efficient prompting techniques, and it can be listed among the examples described in *Chapter 3*.

Now, let's look at some more examples of using ChatGPT for more specific tasks, including text generation, writing assistance, and information retrieval.

Generating text

As a language model, ChatGPT is particularly suited for generating text based on users' instructions. For example, you could ask ChatGPT to generate emails, drafts, or templates that target a specific audience:

executive meeting about OpenAI. Be grabbing maintaining an executive style.

2/2

Email Invitation for Executive Meeting

Subject: Executive Meeting Invitation: Discussing Opportunities with OpenAI

Dear [Customer's Name],

I hope this message finds you well.

I am writing to invite you to an executive meeting at our office to discuss the innovative opportunities and strategic benefits that OpenAI can offer to our business operations. This meeting will provide a platform to explore how leveraging OpenAI's cutting-edge technology can drive efficiency, enhance decision-making, and foster growth within our organization.

Details of the Meeting:

- **Date:** [Insert Date]
- **Time:** [Insert Time]
- **Location:** [Your Office Address]

Your insights and expertise would be highly valuable, and I believe this discussion will pave the way for significant advancements in our collaborative efforts.

Please confirm your availability at your earliest convenience.

Best regards,

Valentina Alto
[Your Position]
[Your Contact Information]

Figure 5.7 – Example of an email generated by ChatGPT

Another example might be asking ChatGPT to create a pitch structure for a presentation you have to prepare (I'll put here only the first slide as an example):

create the structure of a slideshow presentation about ChatGPT. Provide me with an agenda and specify the content of each slide. It should have 3 slides.

Slideshow Presentation on ChatGPT

Agenda:

1. **Introduction to ChatGPT**
2. **Capabilities and Applications**
3. **Future Developments and Ethical Considerations**

Slide 1: Introduction to ChatGPT

Title: What is ChatGPT?

Content:

- **Overview:** ChatGPT is an advanced language model developed by OpenAI, designed to understand and generate human-like text based on the input it receives.
- **Development:** It is built on the GPT-4 architecture, leveraging vast amounts of data and sophisticated algorithms to produce coherent and contextually relevant responses.
- **Purpose:** Its primary aim is to assist with a wide range of tasks including drafting emails, writing code, creating content, and more.

Visuals:

- OpenAI logo
- Image or diagram of the GPT-4 architecture

Figure 5.8 – Slideshow agenda and structure generated by ChatGPT

You can also generate blog posts or articles about trending topics this way. Here is an example:

generate a short blog post about OpenAI DALLE model. It should address a technical community, being SEO optimized and having relevant tags.

2/2

Exploring the Power of OpenAI's DALL-E Model

Introduction

The landscape of AI-generated art has been revolutionized by OpenAI's DALL-E model, an impressive neural network that creates images from textual descriptions. This blog post delves into the technical prowess of DALL-E, its applications, and why it's a significant advancement in the AI community.

What is DALL-E?

DALL-E is a variant of the GPT-3 model, specifically trained to generate images from text inputs. Combining "DALL-E" and "WALL-E," the model epitomizes the fusion of creativity and engineering. DALL-E can interpret detailed text descriptions and produce corresponding visual outputs, making it a groundbreaking tool for various applications in AI and beyond.

Technical Breakdown

Architecture:

- DALL-E utilizes the Transformer architecture, similar to GPT-3, with modifications to handle image data.
- It employs a VQ-VAE-2 (Vector Quantized Variational Autoencoder) to encode and decode images, bridging the gap between textual input and image output.

Training Data:

- The model was trained on a diverse dataset of text-image pairs, enabling it to understand and synthesize complex visual scenes from natural language descriptions.

Capabilities:

- **Creative Design:** DALL-E can generate unique and diverse images from abstract concepts, aiding in creative industries like graphic design and advertising.
- **Prototyping:** Designers and engineers can quickly prototype visual ideas by simply describing them in text.
- **Educational Tools:** The model can be used in educational settings to create illustrative content, making learning more interactive and engaging.

Applications and Use Cases

1. **Art and Design:** DALL-E's ability to create high-quality images from textual descriptions opens up new possibilities for artists and designers to experiment with novel ideas.
2. **Advertising:** Marketers can use DALL-E to generate unique visuals for campaigns, tailoring images to specific themes or concepts without the need for extensive graphic design skills.
3. **Content Creation:** Writers and content creators can enhance their work with custom images that accurately reflect their narrative, improving audience engagement.

Ethical Considerations

While DALL-E represents a significant technological leap, it also raises ethical questions about the use of AI in creative fields. Issues such as the potential for misuse, the authenticity of AI-generated art, and the impact on traditional artists need careful consideration. OpenAI emphasizes responsible deployment and usage of DALL-E to mitigate these concerns.

Conclusion

OpenAI's DALL-E model stands at the forefront of AI-generated art, blending sophisticated neural network architecture with creative potential. For the technical community, it presents an exciting opportunity to explore new applications and push the boundaries of what's possible with AI.

Tags

- #OpenAI
- #DALL-E
- #AIGeneratedArt
- #NeuralNetworks
- #MachineLearning

Figure 5.9 – Example of a blog post with relevant tags and SEO keywords generated by ChatGPT

We can even get ChatGPT to reduce the size of the post to make it fit for a tweet. Here is how we can do this:

shrink the above blog post into an X post

🚀 Excited about OpenAI's DALL-E model! This neural network generates images from text descriptions, blending GPT-3 architecture with VQ-VAE-2 for stunning results. Perfect for creative design, prototyping, and educational tools. Ethical considerations are crucial as we explore this tech's potential. Dive into the future of AI-generated art!

#OpenAI #DALLE #AI #MachineLearning #AIGeneratedArt #CreativeAI

Figure 5.10 – ChatGPT shrinks an article into a Twitter post

Finally, ChatGPT can also generate video or theatre scripts, including the scenography and the suggested editing. The following figure shows an example of a theatre dialog including scenography and actors:

generate a script for a theater dialogue about ChatGPT. Include scenography and actors.

Title: "Conversations with ChatGPT"

Scenography

The stage is set in a modern tech office with a minimalist design. There's a large screen at center stage, displaying the ChatGPT interface. On one side of the stage is a desk with a computer, books, and papers scattered around. The opposite side has a comfortable chair and a small coffee table. The lighting is bright but not harsh, creating a welcoming and innovative atmosphere.

Characters

- **Alex**: A tech-savvy individual, curious and enthusiastic about AI.
- **Jordan**: A skeptical but open-minded person, interested in understanding the practical uses of AI.
- **ChatGPT**: Represented by an actor or voiceover, interacting with Alex and Jordan through the screen.

Scene 1: Introduction

(Alex is sitting at the desk, typing on the computer. Jordan enters, looking around curiously.)

Jordan: Hey Alex, what's got you so engrossed today?

Alex: Hey Jordan! I'm just exploring ChatGPT, this amazing AI developed by OpenAI. It's incredible how it can generate human-like text based on the input it receives.

Jordan: (raises an eyebrow) Really? AI that talks like a human? Show me.

Alex: Sure thing. Let's ask it something. (turns to the screen) Hey ChatGPT, can you introduce yourself?

ChatGPT: (voiceover) Hello! I'm ChatGPT, an AI language model created by OpenAI. I can help with various tasks like answering questions, writing content, and even generating creative pieces.

Jordan: (impressed) That's quite something. How does it work?

Figure 5.11 – Theatre dialog with scenography generated by ChatGPT

I only provided one scene out of the four generated by ChatGPT, to keep you in suspense regarding the ending…

Overall, whenever new content needs to be generated from scratch, ChatGPT does a very nice job of providing a first draft, which could act as the starting point for further refinements.

However, ChatGPT can also support pre-existing content by providing writing assistance and translation, as we will see in the next section.

Improving writing skills and translation

Sometimes, rather than generating new content, you might want to revisit an existing piece of text. It this be for style improvement purposes, audience changes, language translation, and so on.

Let’s look at some examples. Imagine that I drafted an email to invite a customer of mine to a webinar. I wrote two short sentences. Here, I want ChatGPT to improve the form and style of this email since the target audience will be executive level:

Hello Dr. Stuart, I want to share an invitation to my webinar about OpenAI. Find attached the invitation. at disposal for clarifications.

Please improve the above email. It should address a C-level audience. It should be concise yet effective.

Improved Email Invitation

Subject: Exclusive Webinar on OpenAI: Innovation and Future Insights

Dear Dr. Stuart,

I hope this message finds you well.

I am pleased to invite you to our upcoming exclusive webinar on OpenAI, where we will delve into the latest advancements and strategic implications of AI technology. This session is tailored for C-level executives, aiming to provide valuable insights that can drive innovation and competitive advantage in your organization.

Webinar Details:

- **Date:** [Insert Date]
- **Time:** [Insert Time]
- **Topic:** Harnessing OpenAI for Business Innovation

Please find the detailed invitation attached. Should you have any questions or require further information, feel free to reach out.

Looking forward to your participation.

Best regards,

[Your Name]
[Your Position]
[Your Contact Information]

Figure 5.12 – Example of an email revisited by ChatGPT to target an executive audience

Now, let’s ask the same thing but with a different target audience:

re-write the above email changing target audience. Now it should target a high-school audience, trying to capture their attention and convince them to attend the webinar.

Improved Email Invitation

Subject: Join Our Exciting Webinar on OpenAI!

Hey there!

We have an awesome opportunity just for you! Join us for an exciting webinar where we'll dive into the world of OpenAI and explore how artificial intelligence is changing the future. Whether you're into tech, curious about AI, or just love learning new things, this is the perfect event for you.

Webinar Details:

- **Date:** [Insert Date]
- **Time:** [Insert Time]
- **Topic:** Discovering the Power of OpenAI

Check out the attached invitation for more details. If you have any questions or need more info, feel free to reach out. Don't miss out on this chance to learn and have fun!

Hope to see you there!

Best,
[Your Name]
[Your Position]
[Your Contact Information]

Figure 5.13 – Example of the same email with a different audience, generated by ChatGPT

ChatGPT can also give you some feedback about your writing style and structure.

Imagine, for example, that you wrote an introduction for an essay titled *The History of Natural Language Processing* and you want some feedback about the writing style and its consistency with the title:

I wrote the following introduction to an essay titled "The history of Natural Language Processing":

Natural Language Processing (NPL) is a field of Artificial Intelligence whose purpose is finding computational methods to interpret human language as it is spoken or written. The idea of NLP goes beyond a mere classification task which could be carried on by ML algorithms or Deep Learning NNs. Indeed, NLP is about interpretation: you want to train your model not only to detect frequent words, to count them or to eliminate some noisy punctuations; you want it to tell you whether the mood of the conversation is positive or negative, whether the content of an e-mail is mere publicity or something important, whether the reviews about thriller books in last years have been good or bad.

Give me a feedback and some suggestions to improve it in less than 200 words.

2/2

Your introduction provides a clear and concise explanation of Natural Language Processing (NLP). Here are a few suggestions to enhance it:

Feedback and Suggestions:

1. **Clarify Acronym:** Correct the acronym for Natural Language Processing from "NPL" to "NLP" to ensure accuracy.
2. **Broaden Context:** Adding a brief historical context can help set the stage for the essay.
3. **Refine Examples:** Including more diverse examples can illustrate the breadth of NLP applications.
4. **Enhance Flow:** Smooth transitions between ideas can improve readability.

Revised Introduction:

Natural Language Processing (NLP) is a pivotal field of Artificial Intelligence focused on developing computational methods to interpret human language in its spoken or written form. Unlike basic classification tasks handled by machine learning algorithms or deep learning neural networks, NLP emphasizes understanding and interpretation. Beyond detecting frequent words, counting occurrences, or removing noisy punctuation, NLP aims to comprehend the nuances of language. For instance, it can determine the sentiment of a conversation, distinguish between promotional emails and important messages, and analyze whether recent thriller book reviews are favorable or not. This capability makes NLP a powerful tool in various applications, from sentiment analysis and spam detection to literary criticism and beyond.

These adjustments provide a more polished and engaging introduction, setting a solid foundation

for your essay on the history of NLP.

Figure 5.15 – Example of ChatGPT giving feedback on an introduction for an essay

As you can see, not only ChatGPT provided me with feedback and tips for the overall essay, but also it gave me a revised introduction which incorporates all of them.

Let's also ask ChatGPT to be more specific regarding its suggestion of "Refining Examples":

what do you mean with refine examples? can you generate one refined example with respect to my intro?

2/2

Certainly! Here's a refined example to enhance your introduction:

Original Example:

"...whether the mood of the conversation is positive or negative, whether the content of an e-mail is mere publicity or something important, whether the reviews about thriller books in last years have been good or bad."

Refined Example:

"...whether the mood of a social media post reflects joy or frustration, whether an email contains critical updates or is simply spam, and whether customer reviews on e-commerce platforms express satisfaction or disappointment with a product."

Revised Introduction with Refined Example:

Natural Language Processing (NLP) is a pivotal field of Artificial Intelligence focused on developing computational methods to interpret human language in its spoken or written form. Unlike basic classification tasks handled by machine learning algorithms or deep learning neural networks, NLP emphasizes understanding and interpretation. Beyond detecting frequent words, counting occurrences, or removing noisy punctuation, NLP aims to comprehend the nuances of language. For instance, it can determine whether the mood of a social media post reflects joy or frustration, whether an email contains critical updates or is simply spam, and whether customer reviews on e-commerce platforms express satisfaction or disappointment with a product. This capability makes NLP a powerful tool in various applications, from sentiment analysis and spam detection to market research and customer service improvement.

This refined example provides more diverse and relatable scenarios, illustrating the broad applicability and importance of NLP.

Figure 5.16 – Example of ChatGPT elaborating on something it mentioned

I'm also interested in knowing whether my introduction was consistent with the title or whether I'm taking the wrong direction:

is my introduction consistent with the title?

Your introduction is well-written but could be more consistent with the title, "The History of Natural Language Processing." The current introduction focuses more on explaining what NLP is and its applications rather than its historical development. To better align with the title, you might consider briefly mentioning key milestones or figures in the history of NLP in your introduction.

Figure 5.17 – ChatGPT provides feedback about the consistency of the introduction with the title

I was impressed by this last one. ChatGPT was smart enough to see that there was no specific mention of the history of NLP in my introduction. Nevertheless, it sets up the expectation about that topic to be treated later on. This means that ChatGPT also has expertise in terms of how an essay should be structured and it was very precise in applying its judgment, knowing that it was just an introduction.

Let's now unveil the last ChatGPT skill of the Chapter. In fact, ChatGPT is also an excellent tool for translation. It knows at least 95 languages (if you have doubts about whether your language is supported, you can always ask ChatGPT directly). Here, however, there is a consideration that might arise: what is the added value of ChatGPT for translation when we already have cutting-edge tools such as Google Translate?

To answer this question, we have to consider some key differentiators and how we can leverage ChatGPT's embedded translations capabilities:

- ChatGPT can capture the intent. This means that you could also bypass the translation phase since it is something that ChatGPT can do in the backend. For example, if you write a

prompt to produce a social media post in French, you could write that prompt in any language you want – ChatGPT will automatically detect it (without the need to specify it in advance) and understand your intent:

generate a LinkedIn post about GenAI in French

Post LinkedIn en Français sur l'IA Générative

🌏 **Exploration de l'IA Générative : Révolutionner la Créativité et l'Innovation**

L'intelligence artificielle générative (GenAI) est en train de transformer radicalement le paysage technologique et créatif. Grâce à des modèles avancés comme GPT-4 de OpenAI, nous assistons à une nouvelle ère où les machines peuvent non seulement comprendre, mais aussi créer du contenu original et significatif.

✨ **Applications Pratiques :**

- **Création de Contenu** : Rédaction d'articles, génération de scripts et production de contenu marketing.
- **Design et Arts Visuels** : Génération d'images, conception graphique et création artistique assistée par l'IA.
- **Interaction Client** : Chatbots avancés et assistants virtuels qui comprennent et répondent de manière plus humaine.

🚀 **Pourquoi c'est Important :**
L'IA Générative permet aux entreprises d'innover plus rapidement, de personnaliser l'expérience utilisateur et d'améliorer l'efficacité opérationnelle. Elle ouvre également de nouvelles opportunités pour les créatifs et les entrepreneurs, en leur donnant des outils puissants pour réaliser leurs visions.

💡 **À l'Avenir :**
Avec l'évolution rapide des technologies de l'IA, il est essentiel de comprendre et d'explorer les possibilités offertes par l'IA générative. Suivez-nous pour rester informé des dernières avancées et découvrez comment intégrer cette technologie dans vos projets.

#Innovation #Technologie #IA #IntelligenceArtificielle #GenAI #Créativité #TransformationDigitale

Figure 5.18 – Example of ChatGPT generating an output in a language that is different from the input

- ChatGPT can capture the more refined meaning of slang or idioms. This allows for a translation that is not literal so that it can preserve the underlying meaning. Namely, let's consider the British expression It's not my cup of tea, which indicates that something is not to one's liking or preference. Let's ask both ChatGPT and Google Translate to translate it into Italian:

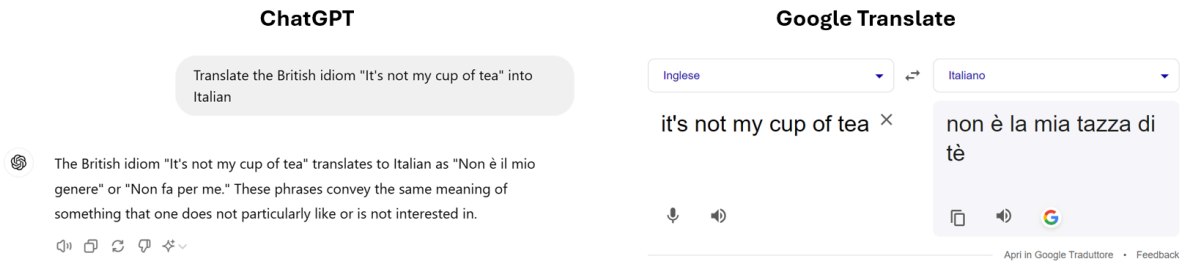


Figure 5.19 – Comparison between ChatGPT and Google Translate while translating from English into Italian

As you can see, ChatGPT can provide several Italian idioms that are equivalent to the original one, also in their slang format. On the other hand, Google Translate performed a literal translation, leaving behind the real meaning of the idiom.

- As with any other task, you can always provide context to ChatGPT. So, if you want your translation to have a specific slang or style, you can always specify it in the prompt. Or, even funnier, you can ask ChatGPT to translate your prompt with a sarcastic touch:

Translate in English the following text with a very sarcastic style.

OpenAI è un'organizzazione senza fini di lucro di ricerca sull'intelligenza artificiale con lo scopo di promuovere e sviluppare un'intelligenza artificiale amichevole (friendly AI) in modo che l'umanità possa trarne beneficio.

Fondata nel tardo 2015, l'organizzazione, con sede a San Francisco, ha lo scopo di "collaborare liberamente" con altre istituzioni e ricercatori rendendo i suoi brevetti e ricerche aperti al pubblico.

I fondatori (tra cui Elon Musk e Sam Altman) sono stati motivati in parte dai rischi esistenziali derivanti dall'intelligenza artificiale generale.

3/3

Sarcastic Translation

OpenAI is a non-profit organization dedicated to AI research with the noble goal of promoting and developing "friendly AI" so that humanity can, of course, benefit from it.

Founded in late 2015, this organization, based in San Francisco, aims to "freely collaborate" with other institutions and researchers by making its patents and research publicly accessible. How generous!

The founders, including none other than Elon Musk and Sam Altman, were partly motivated by the existential risks posed by general artificial intelligence. Because what could go wrong with creating super-intelligent machines?

Figure 5.20 – Example of ChatGPT translating a prompt with a sarcastic touch. The original content of the prompt was taken from OpenAI's Wikipedia page: https://it.wikipedia.org/wiki/OpenAI

All these scenarios highlight one of the key killing features of ChatGPT and OpenAI models in general. Since they represent the manifestation of what OpenAI defined as **Artificial General Intelligence** (**AGI**), they are not meant to be specialized (that is,

constrained) on a single task. On the contrary, they are meant to serve multiple scenarios dynamically so that you can address a wide range of use cases with a single model.

In conclusion, ChatGPT is able not only to generate new text but also to manipulate existing material to tailor it to your needs. It has also proven to be very precise at translating between languages, also keeping the jargon and language-specific expressions intact.

In the next section, we will see how ChatGPT can assist us in retrieving information and competitive intelligence.

Quick information retrieval and competitive intelligence

Information retrieval and competitive intelligence are yet other fields where ChatGPT is a game-changer. The very first example of how ChatGPT can retrieve information is the most popular way it is used right now: as a search engine. Every time we ask ChatGPT something, it can retrieve information from its knowledge base and reframe it in an original way.

One example involves asking ChatGPT to provide a quick summary or review of a book we might be interested in reading:

generate a quick summary (without spoiler of the ending!) of the book "Harry Potter and the Prisoner of Azkaban. Generate also a review of this book.

Summary of "Harry Potter and the Prisoner of Azkaban"

In "Harry Potter and the Prisoner of Azkaban," Harry returns to Hogwarts for his third year, where he learns that the notorious Sirius Black has escaped from Azkaban prison. Black is believed to be after Harry, adding to the tension as the school is guarded by Dementors, sinister creatures that suck the happiness out of anyone near them. Harry discovers more about his past and his parents' connection to Sirius Black. Alongside Ron and Hermione, Harry unravels secrets and encounters new magical creatures, including the shape-shifting Boggart and the enigmatic Hippogriff, Buckbeak. The trio's journey reveals the truth behind Sirius Black's escape and his true intentions, leading to a thrilling adventure filled with unexpected twists.

Review of "Harry Potter and the Prisoner of Azkaban"

"Harry Potter and the Prisoner of Azkaban" is a captivating addition to J.K. Rowling's beloved series. The book masterfully blends elements of mystery, adventure, and the supernatural, keeping readers engrossed from start to finish. Rowling's writing matures with her characters, delving deeper into the darker aspects of the wizarding world while maintaining the charm and wit that define the series. The introduction of Sirius Black and the complex backstory enrich the plot, adding layers of depth to Harry's history. The novel also stands out for its exploration of themes like friendship, loyalty, and the fight against prejudice. The imaginative use of time travel and the introduction of the Dementors provide fresh, thrilling elements to the narrative. Overall, "The Prisoner of Azkaban" is a compelling and richly detailed story that continues to enchant readers of all ages.

Figure 5.21 – Example of ChatGPT providing a summary and review of a book

Alternatively, we could ask for some suggestions for a new book we wish to read based on our preferences:

list me 5 books I should read if I liked a lot the Harry Potter series. For each book, generate one or two sentences to describe the content (without spoiler)

5 Books to Read If You Loved the Harry Potter Series

1. **"Percy Jackson & The Olympians: The Lightning Thief" by Rick Riordan**
   - Follow the adventures of Percy Jackson, a modern-day demigod who discovers his true identity and sets out on a quest to prevent a war among the Greek gods. The series blends mythology with modern settings, offering a thrilling and humorous ride.
2. **"The Chronicles of Narnia: The Lion, the Witch and the Wardrobe" by C.S. Lewis**
   - Enter the magical world of Narnia, where four siblings step through a wardrobe into a land of talking animals, mystical creatures, and an epic battle between good and evil, led by the noble lion Aslan.
3. **"The Golden Compass" by Philip Pullman**
   - Join Lyra Belacqua on a journey through parallel worlds, where she encounters armored bears, witches, and a mysterious substance called Dust. This richly imaginative story explores deep philosophical themes and features a strong, young heroine.
4. **"The Hobbit" by J.R.R. Tolkien**
   - Accompany Bilbo Baggins on an unexpected adventure filled with dwarves, dragons, and treasures. This prelude to the "Lord of the Rings" series offers a delightful tale of bravery, friendship, and the discovery of inner strength.
5. **"Eragon" by Christopher Paolini**
   - Follow the journey of Eragon, a young farm boy who finds a mysterious blue stone that hatches into a dragon. As a Dragon Rider, Eragon is thrust into a world of magic, power struggles, and epic battles, discovering his destiny along the way.

Figure 5.22 – Example of ChatGPT recommending a list of books, given my preferences

Furthermore, if we design the prompt with more specific information, ChatGPT can serve as a tool for pointing us toward the right references for our research or studies.

Namely, you might want to quickly retrieve some background references about a topic you want to learn more about – for example, feedforward neural networks. Something you might ask ChatGPT is to point you to some websites or papers where this topic is widely treated:

I want to learn more about Feedforward Neural Network. List 3 relevant papers or website I can start from in order to study the topic

To start learning about Feedforward Neural Networks, you can refer to the following resources:

1. **"Neural Networks and Deep Learning" by Michael Nielsen**
   This online book provides a comprehensive introduction to neural networks, including feedforward neural networks. It covers the fundamental concepts, mathematical foundations, and practical implementations.

   - Read it online

2. **"Understanding Machine Learning: From Theory to Algorithms" by Shai Shalev-Shwartz and Shai Ben-David**
   This book includes a detailed chapter on neural networks, discussing the structure, training, and theoretical aspects of feedforward networks.

   - Read it on Cambridge University Press

3. **"Gradient-Based Learning Applied to Document Recognition" by Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner (1998)**
   This seminal paper discusses the use of gradient-based learning algorithms for training neural networks and provides insights into the architecture and application of feedforward neural networks.

   - Read the paper

These resources will give you a solid foundation in the theory and practice of feedforward neural networks, helping you understand their structure, functionality, and applications.

Figure 5.23 – Example of ChatGPT listing relevant references

As you can see, ChatGPT was able to provide me with relevant references to start studying the topic. However, it could go even further in terms of competitive intelligence.

Let’s consider I’m writing a book titled Introduction to *Convolutional Neural Networks – an Implementation with Python*. I want to do some research about the potential competitors in the market. The first thing I want to investigate is whether there are already some competitive titles around, so I can ask ChatGPT to generate a list of existing books with the same content:

I want to write a book titled “Introduction to Convolutional Neural Network: an implementation with Python”. Do a web research and provide me with a list of 3 existing books with a competitive content along with their web references.

4/4

Searched 6 sites

Here are three books that cover Convolutional Neural Networks (CNNs) with Python, which could serve as competitive content for your book "Introduction to Convolutional Neural Network: An Implementation with Python":

1. **"Deep Learning with Python: A Hands-On Introduction" by Nikhil Ketkar**
   - This book provides a thorough introduction to deep learning, including detailed chapters on convolutional neural networks. It covers the essential concepts, the convolution operation, and practical implementations using Python. The hands-on approach makes it accessible for beginners and intermediate learners.
   - Read more on O'Reilly
2. **"Hands-On Convolutional Neural Networks with TensorFlow" by Iffat Zafar, Giounona Tzanidou, Richard Burton, Nimesh Patel, and Leonardo Araujo**
   - This book is a practical guide to implementing convolutional neural networks using TensorFlow. It covers various CNN architectures, their applications in computer vision, and real-world problem-solving techniques. The book includes numerous examples and exercises to help readers solidify their understanding.
   - Find it on O'Reilly
3. **"A Guide to Convolutional Neural Networks for Computer Vision" by Salman Khan, Hossein Rahmani, Syed Afaq Ali Shah, and Mohammed Bennamoun**
   - This guide provides an in-depth look at CNNs specifically for computer vision tasks. It includes discussions on the basics of neural networks, detailed explanations of CNN architectures, and practical applications in image classification and object detection. The book also reviews state-of-the-art models and techniques used in the field.
   - Explore on SpringerLink

Figure 5.24 – Example of ChatGPT providing a list of competitive books

You can also ask for feedback in terms of the saturation of the market you want to publish in:

give me a concise and short feedback about the saturation of the market with regards to the content of my book

3/3

The market for books on Convolutional Neural Networks (CNNs) and their implementation with Python is becoming increasingly saturated, with numerous comprehensive guides already available. Titles such as "Deep Learning with Python" by Nikhil Ketkar, "Hands-On Convolutional Neural Networks with TensorFlow" by Iffat Zafar and others, and "A Guide to Convolutional Neural Networks for Computer Vision" by Salman Khan et al. are well-regarded resources that cover similar content. This saturation suggests that to stand out, your book will need a unique angle or significant value addition, such as novel insights, more practical examples, or an emphasis on a specific application or audience.

For further details, you can refer to:

- Deep Learning with Python: A Hands-On Introduction
- Hands-On Convolutional Neural Networks with TensorFlow
- A Guide to Convolutional Neural Networks for Computer Vision

Figure 5.25 – ChatGPT advising about how to be competitive in the market

Finally, let's ask ChatGPT to be more precise about what I should do to be competitive in the market where I will operate:

elaborate some examples of how my book could stand out in a competitive environment

To make your book "Introduction to Convolutional Neural Network: An Implementation with Python" stand out in a competitive market, consider incorporating the following unique elements:

1. Focus on Real-World Applications

- **Industry-Specific Case Studies:** Include detailed case studies from various industries such as healthcare, finance, and autonomous driving. Show how CNNs solve specific real-world problems.
- **Practical Projects:** Provide step-by-step guides for real-world projects, such as building a facial recognition system, a medical image classifier, or an autonomous vehicle vision system.

2. Interactive Learning Resources

- **Code Examples and Exercises:** Offer extensive, well-documented code examples and exercises. Use interactive Jupyter notebooks that readers can run and modify.
- **Online Companion Resources:** Create a companion website with additional resources such as video tutorials, datasets, and a forum for discussion and Q&A.

3. Focus on Explainability and Ethics

- **Explainability Techniques:** Dedicate a section to explainability in CNNs, covering techniques like Grad-CAM, SHAP, and LIME, to help users understand and trust model outputs.
- **Ethical Considerations:** Discuss ethical considerations in AI, such as bias in datasets, privacy concerns, and the importance of transparent AI practices.

4. Cutting-Edge Topics and Research

- **Latest Research Insights:** Include summaries of the latest research papers and advancements in CNNs. Highlight trends like transformers in vision, few-shot learning, and unsupervised learning techniques.
- **Interviews with Experts:** Feature interviews with leading researchers and practitioners in the field of CNNs to provide insider insights and future directions.

5. Beginner to Advanced Pathway

- **Structured Learning Path:** Design the book to cater to a wide range of readers, from beginners to advanced practitioners. Start with fundamental concepts and progressively cover more advanced topics.
- **Hands-On Implementation:** Ensure that each chapter includes hands-on implementation sections, guiding readers from theory to practice seamlessly.

6. Community and Collaboration

- **Interactive Community:** Encourage readers to join a community platform where they can share their projects, get feedback, and collaborate on CNN-related topics.
- **Collaborative Projects:** Include chapters that encourage readers to participate in collaborative projects or competitions, fostering a sense of community and real-world experience.

7. Multimedia Integration

- **Video Tutorials:** Integrate QR codes linking to video tutorials that walk through complex concepts and coding exercises.
- **Visualization Tools:** Use visual aids and interactive tools to demonstrate how CNNs process and interpret images, making the learning process more intuitive.

Figure 5.26 – Example of how ChatGPT can suggest improvements regarding your book content to make it stand out

ChatGPT was pretty good at listing some good tips to make my book unique.

Overall, ChatGPT can be a valuable assistant for information retrieval and competitive intelligence. However, it is important to remember the knowledge base cut-off is 2021: this means that, whenever we need to retrieve real-time information, or while making a competitive market analysis for today, we might not be able to rely on ChatGPT.

Nevertheless, this tool still provides excellent suggestions and best practices that can be applied, regardless of the knowledge base cut-off.

Summary

All the examples we saw in this chapter were modest representations of what you can achieve with ChatGPT to boost your productivity. These small hacks can greatly assist you with activities that might be repetitive (answering emails with a similar template rather than writing a daily routine) or onerous (such as searching for background documentation or competitive intelligence).

In the next chapter, we are going to dive deeper into three main domains where ChatGPT is changing the game – development, marketing, and research.

5 Developing the Future with ChatGPT

Join our book community on Discord

https://packt.link/EarlyAccess

In this chapter, we will discuss how developers can leverage ChatGPT. The chapter focuses on the main use cases ChatGPT addresses in the domain of developers, including code review and optimization, documentation generation, and code generation. The chapter will provide examples and enable you to try the prompts on your own.

After a general introduction about the reasons why developers should leverage ChatGPT as a daily assistant, we will focus on ChatGPT and how it can do the following:

- Why ChatGPT for developers?
- Generate, optimize, and debug code
- Generate code-related documentation and debug your code
- Explain **machine learning** (**ML**) models to help data scientists and business users with model interpretability
- Translate different programming languages

By the end of this chapter, you will be able to leverage ChatGPT for coding activities and use it as an assistant for your coding productivity.

Why ChatGPT for developers?

Personally, I believe that one of the most mind-blowing capabilities of ChatGPT is that of dealing with code. Of any type. We've already seen in previous chapters some examples of ChatGPT generating Python code. However, ChatGPT capabilities for developers go way beyond that example. It can be a daily assistant for code generation, explanation, and debugging.

Among the most popular languages, we can certainly mention Python, JavaScript, SQL, and C#. However, ChatGPT covers a wide range of languages, as disclosed by itself:

List all the programming languages you can understand and generate

< 4/4 >

I can understand and generate content in the following programming languages:

1. Python
2. JavaScript
3. Java
4. C#
5. C++
6. C
7. Ruby
8. PHP
9. Swift
10. Kotlin
11. R
12. TypeScript
13. Go
14. Rust
15. MATLAB
16. Perl
17. SQL
18. HTML/CSS (for web development)
19. Shell scripting (Bash)
20. PowerShell
21. Scala
22. Haskell
23. Lua
24. Objective-C
25. Dart
26. Groovy
27. Julia
28. Visual Basic
29. Assembly

If you need help with any of these languages or anything related to them, feel free to ask!

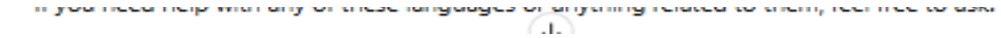


Figure 6.1 – ChatGPT lists the programming languages it is able to understand and generate

Whether you are a backend/frontend developer, a data scientist, or a data engineer, whenever you work with a programming language, ChatGPT can be a game changer, and we will see how in the several examples in the next sections.

From the next section onward, we will dive deeper into concrete examples of what ChatGPT can achieve when working with code. We will see end-to-end use cases covering different domains so that we will get familiar with using ChatGPT as a code assistant.

Generating, optimizing, and debugging code

The primary capability you should leverage is ChatGPT code generation. How many times have you been looking for a pre-built piece of code to start from? Generating the utils functions, sample datasets, SQL schemas, and so on? ChatGPT is able to generate code based on input in natural language:

generate a python function that add a line to all my csv files saved within "my_folder". The line should be the name of the csv file.

Certainly! Below is a Python function that adds a line containing the name of each CSV file to the beginning of all CSV files saved within a folder named "my_folder":

python Copy code

```python
import os
import csv

def add_filename_to_csv(folder_path):
    # Get a list of all CSV files in the folder
    csv_files = [f for f in os.listdir(folder_path) if f.endswith('.csv')]

    for csv_file in csv_files:
        file_path = os.path.join(folder_path, csv_file)

        # Read the original content of the CSV file
        with open(file_path, 'r', newline='', encoding='utf-8') as file:
            reader = list(csv.reader(file))

        # Add the filename to the beginning of the CSV content
        new_content = [[csv_file]] + reader

        # Write the new content back to the CSV file
        with open(file_path, 'w', newline='', encoding='utf-8') as file:
            writer = csv.writer(file)
            writer.writerows(new_content)

# Example usage
folder_path = 'my_folder'
add_filename_to_csv(folder_path)
```

Make sure to replace `'my_folder'` with the actual path to your folder containing the CSV files. This function will read each CSV file, prepend the filename as the first line, and save the updated content back to the same file.

Figure 6.2 – Example of ChatGPT generating a Python function to write into CSV files

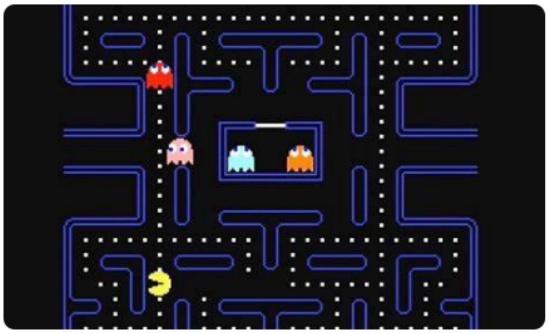
As you can see, not only was ChatGPT able to generate the function, but also it was able to explain what the function does, how to use it, and what to substitute with generic placeholders such as

```
my_folder.
```

Now let's raise the difficulty bar. If ChatGPT is capable of generating a Python function, could it generate an entire Video Game as well? Let's try. What I want to do, is providing ChatGPT with an illustration of the type of game I want to develop, and ask it to replicate it with code. The following is an illustration of my desired game (can you guess the name?):

Figure 1: Illustration of the game Pac-Man.

Now let's ask ChatGPT to reproduce it:



Generate the HTML, JS and CSS code to reproduce the pac-man game. It should look like the illustration provided.

Creating a complete Pac-Man game with HTML, JavaScript, and CSS involves a fair amount of code. Below is a simplified version to help you get started. This example includes basic HTML, CSS, and JavaScript to create a playable Pac-Man game. For a full-featured game, you would need to add more functionality such as collision detection, ghost AI, and scoring.

HTML (index.html)

html Copy code

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Pac-Man Game</title>
    <link rel="stylesheet" href="styles.css">
</head>
<body>
    <canvas id="gameCanvas" width="500" height="500"></canvas>
    <script src="script.js"></script>
</body>
</html>
```

CSS (styles.css)

Figure 2: Example of ChatGPT generating a HTML, CSS and JS code.

As disclaimed by ChatGPT, the full game required a lot of code, however, let's see how the generated code works insofar (to run the code, I used the online tool codepen.io):

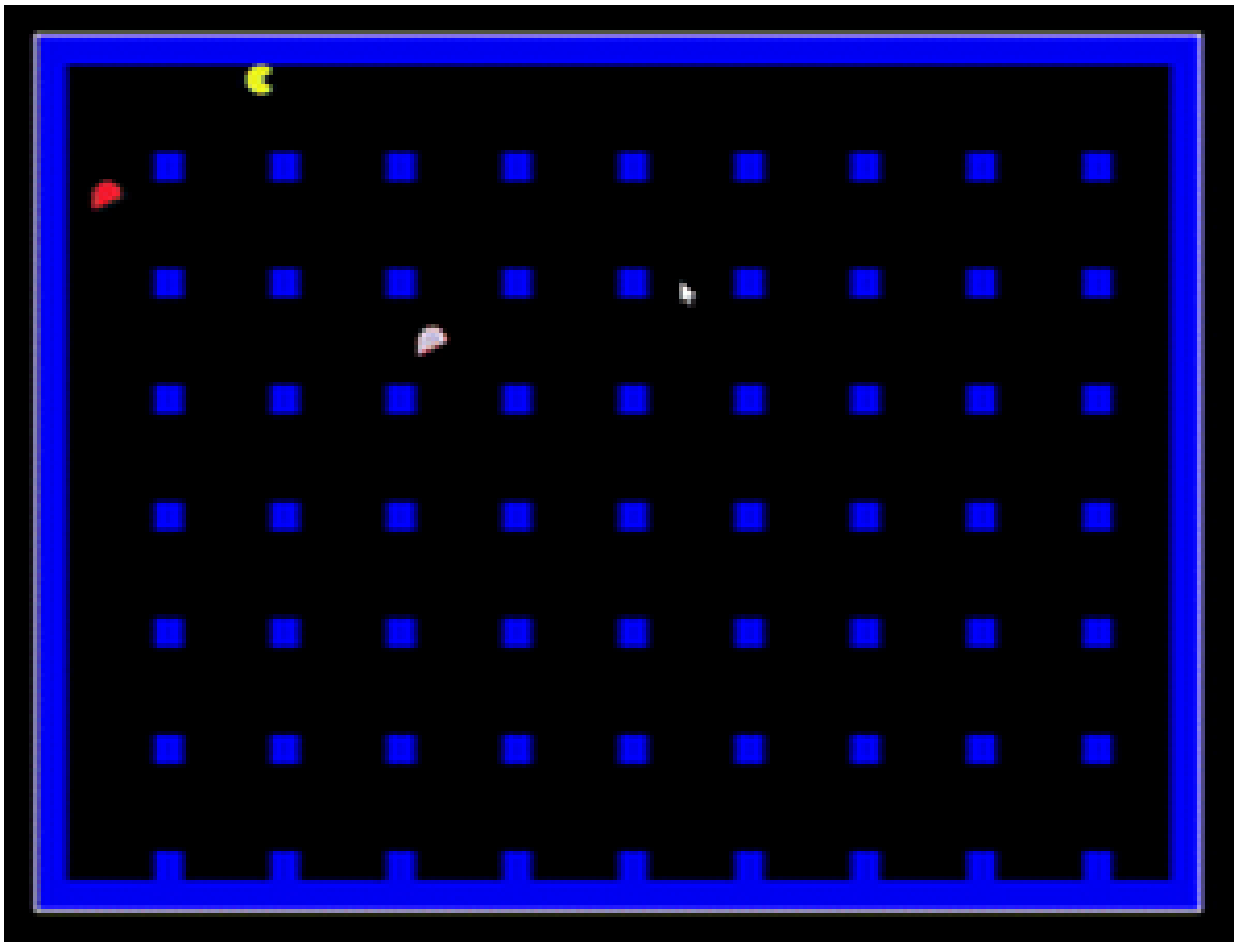


Figure 3: Pac-Man game generated by ChatGPT.

As you can see, the draft product looks already similar to what I'm aiming for! This is an example on how Generative AI can help you overcoming the "curse" of starting from scratch: in fact, starting from a whitepaper can be sometimes blocking, while having a draft product to start from can not only speed up the overall process, but also stimulate creativity and improve the quality of the result.

ChatGPT can also be a great assistant for code optimization. In fact, it might save us some running time or compute power to make optimized scripts starting from our input. This capability might be compared, in the domain of natural language, to the writing assistance feature we saw in Chapter 5 in the Improving writing skills and translation section.

For example, imagine you want to create a list of odd numbers starting from another list. To achieve the result, you wrote the following Python script (for the purpose of this exercise, I will also track the execution time with the timeit and datetime libraries):

```
from timeit import default_timer as timer from datetime import timedelta
start = timer()
elements = list(range(1_000_000)) data = []
for el in elements: if not el % 2:
# if odd number data.append(el)
end = timer() print(timedelta(seconds=end-start))
```

Let's see how long it takes to run:

```python
from timeit import default_timer as timer
from datetime import timedelta

start = timer()
elements = list(range(1_000_000))
data = []
for el in elements:
    if not el % 2:
        data.append(el)

end = timer()
print(timedelta(seconds=end-start))
```

✓ 0.0s

```
0:00:00.115022
```

Figure 4: Speed of execution of a Python function.

The execution time was 00.115022 seconds. What happens if we ask ChatGPT to optimize this script?

I wrote the following script to filter a list only for odd numbers.

```
elements = list(range(1_000_000))
data = []
for el in elements:
  if el % 2 != 0:
    data.append(el)
```

Can you create a script to achieve the same result in lower execution time?

Certainly! You can achieve the same result in a more efficient way using list comprehensions, which are generally faster than a for-loop with an append operation in Python.

Here is the optimized script using a list comprehension:

python Copy code

```python
elements = list(range(1_000_000))
data = [el for el in elements if el % 2 != 0]
```

This code should run faster because list comprehensions are optimized for performance in Python.

If you want an even faster approach, you can use the `filter` function combined with a lambda function, which can sometimes offer better performance:

python Copy code

```python
elements = list(range(1_000_000))
data = list(filter(lambda el: el % 2 != 0, elements))
```

Both of these methods will yield the same result but should execute in less time compared to the original script.

Figure 6.6 – ChatGPT generating optimized alternatives to a Python script

ChatGPT provided me with two examples to achieve the same results with lower execution time.

Let's test both of them in a Jupyter Notebook:

```
2023-03-25 11:27:10.270 Uncaught app exception Traceback (most recent call last):
File "C:\Users\vaalt\Anaconda3\lib\site-packages\streamlit\runtime\ scriptrunner\script_runner.p
```

```
exec(code, module. dict )
File "C:\Users\vaalt\OneDrive\Desktop\medium articles\llm.py", line 129, in <module>
user_input = get_text()
File "C:\Users\vaalt\OneDrive\Desktop\medium articles\llm.py", line 50, in get_text
input_text = st.text_input("You: ", st.session_state['input'], key='input', placeholder = 'Your A
File "C:\Users\vaalt\Anaconda3\lib\site-packages\streamlit\runtime\ metrics_util.py", line 311, i
result = non_optional_func(*args, **kwargs)
File "C:\Users\vaalt\Anaconda3\lib\site-packages\streamlit\elements\ text_widgets.py", line 174,
return self._text_input(
```

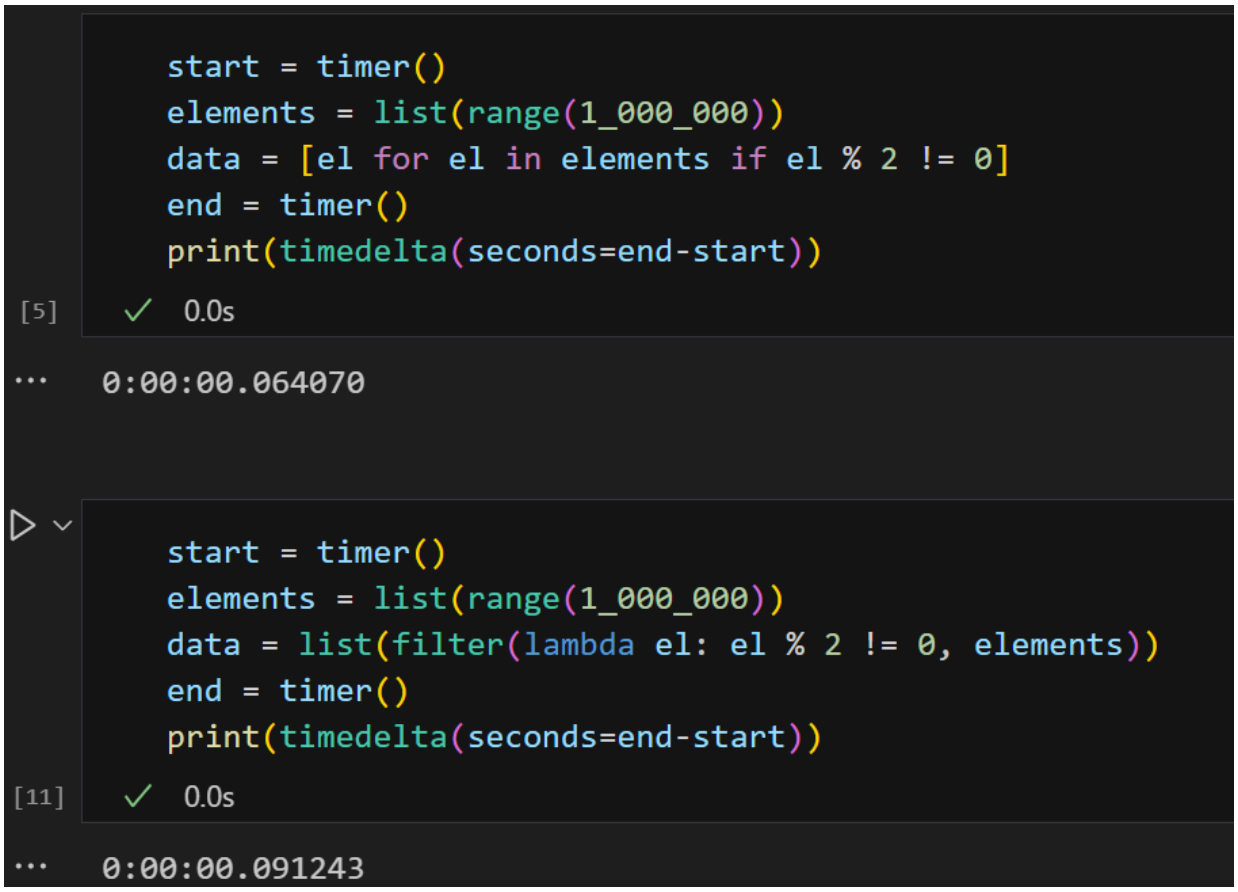


Figure 6.7 – Speed of execution of two alternative functions generated by ChatGPT.

As you can see, both methods lead to a great reduction in time, respectively of 44.30% and 20.68%.

On top of code generation and optimization, ChatGPT can also be leveraged for error explanation and debugging. Sometimes, errors are difficult to interpret; hence a natural language explanation can be useful to identify the problem and drive you toward the solution.

For example, while running a .py file from my command line, I get the following error:

```
File "C:\Users\vaalt\Anaconda3\lib\site-packages\streamlit\elements\ text_widgets.py", line 266,
text_input_proto.value = widget_state.value
TypeError: [] has type list, but expected one of: bytes, Unicode
```

Let's see whether ChatGPT is able to let me understand the nature of the error. To do so, I simply provide ChatGPT with the text of the error and ask it to give me an explanation.

Error Message Breakdown

1. **File Paths and Line Numbers**:
   - The error occurs in your script at `C:\Users\vaalt\OneDrive\Desktop\medium articles\llm.py` on line 129.
   - The error originates from a call to `get_text()`, specifically from `input_text = st.text_input(...)` on line 50 of the same script.
2. **TypeError Explanation**:
   - `TypeError: [] has type list, but expected one of: bytes, Unicode`
   - This error indicates that a list type (`[]`) is being used where a string (either `bytes` or `Unicode`) is expected.

Relevant Code Snippet

The error occurs in the following code (based on your provided traceback):

python Copy code

```python
input_text = st.text_input(
    "You: ",
    st.session_state['input'],
    key='input',
    placeholder='Your AI assistant here! Ask me anything...',
    label_visibility='hidden'
)
```

Cause of the Error

The `text_input` function in Streamlit expects the second argument to be a string (either `bytes` or `Unicode`). However, it appears that `st.session_state['input']` is currently a list.

Solution

You need to ensure that `st.session_state['input']` is a string before passing it to `st.text_input`. Here are the steps to fix this:

1. **Initialize `st.session_state['input']`**: Make sure `st.session_state['input']` is properly initialized as a string.
2. **Check the Initialization**: Ensure the session state is not being set to a list elsewhere in your code.

Here is an updated version of your code:

python Copy code

```python
import streamlit as st

# Initialize the session state if it does not exist
if 'input' not in st.session_state:
    st.session_state['input'] = ''

def get_text():
    # Ensure st.session_state['input'] is a string before passing it to text_input
    if isinstance(st.session_state['input'], list):
        st.session_state['input'] = ''.join(st.session_state['input'])  # Convert list to
    input_text = st.text_input(
        "You: ",
        st.session_state['input'],
        key='input',
        placeholder='Your AI assistant here! Ask me anything...',
        label_visibility='hidden'
    )
    return input_text

# Rest of your code
user_input = get_text()
```

Explanation

1. **Initialization Check**: The script checks if `st.session_state['input']` is not already initialized and sets it to an empty string if it is not.
2. **Type Conversion**: Before calling `st.text_input`, the script checks if `st.session_state['input']` is a list and converts it to a string if necessary.

By ensuring that `st.session_state['input']` is always a string when passed to `st.text_input`, the `TypeError` should be resolved.

Figure 6.8 – ChatGPT explaining a Python error in natural language

Finally, let's imagine I wrote a function in Python that takes a string as input and returns the same string with an underscore after each letter.

In the preceding example, I was expecting to see the g_p_t_ result; however, it only returned t_

with this code:

```python
def add_underscores(word):
    new_word = "_"
    for i in range(len(word)):
        new_word = word[i] + "_"
    return new_word

phrase = "gpt"
print(add_underscores(phrase))
```

[12] ✓ 0.0s

```
t_
```

Figure 6.9 – Bugged Python function

Let's ask ChatGPT to debug this function for us:

I wrote this python function to add an underscore after each letter of a word.
The expected result for the input word should be w_o_r_d_.

```
def add_underscores(word):
    new_word = "_"
    for i in range(len(word)):
        new_word = word[i] + "_"
    return new_word

phrase = "gpt"
print(add_underscores(phrase))
```

can you debug this code?

Sure! Let's debug the function you provided. The goal is to add an underscore after each letter of a word. Your current implementation has a couple of issues. Specifically, the variable `new_word` is being reassigned on each iteration, which results in only the last character and underscore being stored in `new_word`.

Here's the corrected function:

python Copy code

```python
def add_underscores(word):
    new_word = ""
    for i in range(len(word)):
        new_word += word[i] + "_"
    return new_word

phrase = "gpt"
print(add_underscores(phrase))
```

Explanation

1. **Initialization**: Initialize `new_word` as an empty string.
2. **Concatenation**: Use the `+=` operator to concatenate each character and an underscore to `new_word` within the loop.
3. **Return**: Return the final `new_word` after the loop completes.

Figure 6.10 – Example of ChatGPT debugging a Python function

Impressive, isn't it? Again, ChatGPT provided the correct version of the code, and it helped in the explanation of where the bugs were and why they led to an incorrect result. Let's see whether it works now:

```
def add_underscores(word):
    new_word = ""
    for i in range(len(word)):
        new_word += word[i] + "_"
    return new_word

phrase = "gpt"
print(add_underscores(phrase))
```

[13] ✓ 0.0s

```
g_p_t_
```

Figure 6.11 – Python function after ChatGPT debugging

Well, it obviously does!

These and many other code-related functionalities could really boost your productivity, shortening the time to perform many tasks.

However, ChatGPT goes beyond pure debugging. Thanks to the incredible language understanding of the GPT model behind, this GenAI tool is able to generate proper documentation alongside the code, as well as explain exactly what a string of code will do, which we will see in the next section.

Generating documentation and code explainability

Whenever working with new applications or projects, it is always good practice to correlate your code with documentation. It might be in the form of a docstring that you can embed in your functions or classes so that others can invoke them directly in the development environment.

For example, let's consider the same function developed in the previous section and let's make it a Python class:

```
class UnderscoreAdder:
    def __init__(self, word):
        self.word = word
    def add_underscores(self):
        new_word = ""
        for i in range(len(self.word)):
            new_word += self.word[i] + "_"
        return new_word
```

We can test it as follows:

```
class UnderscoreAdder:
    def __init__(self, word):
        self.word = word

    def add_underscores(self):
        new_word = ""
        for i in range(len(self.word)):
            new_word += self.word[i] + "_"
        return new_word

# Example usage:
phrase = "gpt"
adder = UnderscoreAdder(phrase)
print(adder.add_underscores())
```

[14] ✓ 0.0s

```
g_p_t_
```

Figure 5: Testing the UnderscoreAdder class.

Now, let's say I want to be able to retrieve the docstring documentation using the `UnderscoreAdder?` convention. By doing so with Python packages, functions, and methods, we have full

documentation of the capabilities of that specific object, as follows (an example with the pandas Python library):

```
import pandas as pd
pd?
```

```
Type:        module
String form: <module 'pandas' from 'C:\\Users\\vaalt\\Anaconda3\\lib\\site-packages\\pandas\\__init__.py'>
File:        c:\users\vaalt\anaconda3\lib\site-packages\pandas\__init__.py
Docstring:
pandas - a powerful data analysis and manipulation library for Python
======================================================================

**pandas** is a Python package providing fast, flexible, and expressive data
structures designed to make working with "relational" or "labeled" data both
easy and intuitive. It aims to be the fundamental high-level building block for
doing practical, **real world** data analysis in Python. Additionally, it has
the broader goal of becoming **the most powerful and flexible open source data
analysis / manipulation tool available in any language**. It is already well on
its way toward this goal.

Main Features
-------------
Here are just a few of the things that pandas does well:

  - Easy handling of missing data in floating point as well as non-floating
    point data.
  - Size mutability: columns can be inserted and deleted from DataFrame and
    higher dimensional objects
  - Automatic and explicit data alignment: objects can be explicitly aligned
    to a set of labels, or the user can simply ignore the labels and let
    `Series`, `DataFrame`, etc. automatically align the data for you in
    computations.
  - Powerful, flexible group by functionality to perform split-apply-combine
    operations on data sets, for both aggregating and transforming data.
  - Make it easy to convert ragged, differently-indexed data in other Python
    and NumPy data structures into DataFrame objects.
  - Intelligent label-based slicing, fancy indexing, and subsetting of large
    data sets.
  - Intuitive merging and joining data sets.
  - Flexible reshaping and pivoting of data sets.
  - Hierarchical labeling of axes (possible to have multiple labels per tick).
  - Robust IO tools for loading data from flat files (CSV and delimited),
    Excel files, databases, and saving/loading data from the ultrafast HDF5
    format.
  - Time series-specific functionality: date range generation and frequency
    conversion, moving window statistics, date shifting and lagging.
```

Figure 6.13 – Example of the pandas library documentation

So, let's now ask ChatGPT to produce the same result for our `UnderscoreAdder` class.

Generate docstring documentation for the following class:

```
class UnderscoreAdder:
    def __init__(self, word):
        self.word = word

    def add_underscores(self):
        new_word = ""
        for i in range(len(self.word)):
            new_word += self.word[i] + "_"
        return new_word
```

Sure! Here's the `UnderscoreAdder` class with docstring documentation added:

python Copy code

```python
class UnderscoreAdder:
    """
    A class to add underscores after each letter of a word.

    Attributes
    ----------
    word : str
        The word to which underscores will be added.

    Methods
    -------
    add_underscores():
        Returns a new string with underscores added after each letter of the word.
    """

    def __init__(self, word):
        """
        Initializes the UnderscoreAdder with a word.

        Parameters
        ----------
        word : str
            The word to which underscores will be added.
        """
        self.word = word

    def add_underscores(self):
        """
        Adds an underscore after each letter of the word.

        Returns
        -------
        str
            A new string with underscores added after each letter of the word.
        """
        new_word = ""
        for i in range(len(self.word)):
            new_word += self.word[i] + "_"
        return new_word

# Example usage:
phrase = "gpt"
adder = UnderscoreAdder(phrase)
print(adder.add_underscores())  # Output: g_p_t_
```

Explanation of Docstrings

1. **Class Docstring**:
   - Provides a brief description of the class and its purpose.
   - Lists the attributes and methods of the class.
2. **`__init__` Method Docstring**:
   - Describes the initialization method and its parameter(s).
3. **`add_underscores` Method Docstring**:
   - Describes what the method does.
   - Specifies the return type and what it returns.

This documentation will help users understand what the class and its methods do, as well as the expected inputs and outputs.

Figure 6.14 – ChatGPT updating the code with documentation

As a result, if we update our class as shown in the preceding code and `UnderscoreAdder`?, we will get the following output:

```
UnderscoreAdder?
[16]  ✓ 0.1s
...  Init signature: UnderscoreAdder(word)
     Docstring:
     A class to add underscores after each letter of a word.

     Attributes
     ----------
     word : str
         The word to which underscores will be added.

     Methods
     -------
     add_underscores():
         Returns a new string with underscores added after each letter of the word.
     Init docstring:
     Initializes the UnderscoreAdder with a word.

     Parameters
     ----------
     word : str
         The word to which underscores will be added.
     Type:           type
     Subclasses:
```

Figure 6.15 – The new UnderscoreAdder class documentation

Finally, ChatGPT can also be leveraged to explain what a script, function, class, or other similar things do in natural language. We have already seen many examples of ChatGPT enriching its code-related response with clear explanations. However, we can boost this capability by asking specific questions in terms of code understanding.

For example, let's ask ChatGPT to explain to us what the following Python script does:

Explain the following code to me:

```
from azureml.opendatasets import NycTlcYellow
from datetime import datetime
from dateutil import parser

start_date = parser.parse('2018-05-01')
end_date = parser.parse('2018-05-07')

nyc_tlc = NycTlcYellow(start_date=start_date, end_date=end_date)
nyc_tlc_df = nyc_tlc.to_pandas_dataframe()
nyc_tlc_df.info()
```

2/2

Sure! Let's go through the code step by step:

1. Import Statements

```python
from azureml.opendatasets import NycTlcYellow
from datetime import datetime
from dateutil import parser
```

- `from azureml.opendatasets import NycTlcYellow`: This line imports the `NycTlcYellow` class from the `azureml.opendatasets` module. `NycTlcYellow` is a dataset class for accessing New York City Taxi and Limousine Commission (TLC) Yellow taxi trip records.
- `from datetime import datetime`: This line imports the `datetime` class from the `datetime` module. This class is used for manipulating dates and times.
- `from dateutil import parser`: This line imports the `parser` module from the `dateutil` package. `parser` is used for parsing date strings into `datetime` objects.

2. Parsing Dates

```python
start_date = parser.parse('2018-05-01')
end_date = parser.parse('2018-05-07')
```

- `parser.parse('2018-05-01')`: This line parses the string `'2018-05-01'` into a `datetime` object representing May 1, 2018.
- `parser.parse('2018-05-07')`: This line parses the string `'2018-05-07'` into a `datetime` object representing May 7, 2018.

3. Accessing the Dataset

```python
nyc_tlc = NycTlcYellow(start_date=start_date, end_date=end_date)
```

- `NycTlcYellow(start_date=start_date, end_date=end_date)`: This line creates an instance of the `NycTlcYellow` class, specifying the start and end dates for the dataset. This instance represents the NYC Yellow taxi trip records within the specified date range (from May 1, 2018, to May 7, 2018).

4. Converting to Pandas DataFrame

```python
nyc_tlc_df = nyc_tlc.to_pandas_dataframe()
```

- `nyc_tlc.to_pandas_dataframe()`: This method converts the NYC TLC Yellow taxi trip records dataset into a Pandas DataFrame. A DataFrame is a 2-dimensional labeled data structure with columns of potentially different types, which is useful for data analysis and manipulation.
- `nyc_tlc_df`: This variable now holds the dataset as a Pandas DataFrame.

5. Displaying Information about the DataFrame

```python
nyc_tlc_df.info()
```

- `nyc_tlc_df.info()`: This method prints a concise summary of the DataFrame, including the number of entries, column names, non-null counts, and data types of each column.

Figure 6.16 – Example of ChatGPT explaining a Python script

Code explainability can also be part of the preceding mentioned documentation, or it can be used among developers who might want to better understand complex code from other teams or (as sometimes happens to me) remember what they wrote some time ago.

Thanks to ChatGPT and the capabilities mentioned in this section, developers can easily keep track of the project life cycle in natural language so that it is easier for both new team members and non-technical users to understand the work done so far.

We will see in the next section how code explainability is a pivotal step for ML model interpretability in data science projects.

Understanding ML model interpretability

Model interpretability refers to the degree of ease with which a human can comprehend the logic behind the ML model's predictions. Essentially, it is the capability to comprehend how a model arrives at its decisions and which variables are contributing to its forecasts.

Let's see an example of model interpretability using a deep learning **convolutional neural network** (**CNN**) for image classification. I built my model in Python using Keras. For this purpose, I will download the CIFAR-10 dataset directly from keras.datasets: it consists of 60,000 32x32 color images (so 3-channels images) in 10 classes (airplane, automobile, bird, cat, deer, dog, frog, horse, ship, and truck), with 6,000 images per class. Here, I will share just the body of the model; you can find all the related code in the book's GitHub repository for data preparation and pre-processing

at https://github.com/PacktPublishing/The-Ultimate-Guide-to-ChatGPT-and-OpenAI/tree/main/Chapter%206%20-%20ChatGPT%20for%20Developers/code:

```
model=tf.keras.Sequential()
model.add(tf.keras.layers.Conv2D(32,kernel_ size=(3,3),activation='relu',input_shape=
(32,32,1)))
model.add(tf.keras.layers.MaxPooling2D(pool_size=(2,2))) model.add(tf.keras.layers.Flatten()) mo
```

The preceding code is made of several layers that perform different actions. I might be interested in having an explanation of the structure of the model as well as the purpose of each layer. Let's ask ChatGPT for some help with that (below you can see an extract of the response):

Adding Layers to the Model

1. **Convolutional Layer**

```python
model.add(tf.keras.layers.Conv2D(32, kernel_size=(3, 3), activation='relu', input_shape=(3
```

- `tf.keras.layers.Conv2D`: This adds a 2D convolutional layer to the model.
  - `32`: The number of filters (output channels) in the convolution.
  - `kernel_size=(3, 3)`: The dimensions of the convolution window.
  - `activation='relu'`: The activation function to use, ReLU (Rectified Linear Unit) in this case, which helps in introducing non-linearity.
  - `input_shape=(32, 32, 1)`: The shape of the input data, where `32x32` is the size of the input image, and `1` indicates that the images are grayscale (single channel).

2. **Max Pooling Layer**

```python
model.add(tf.keras.layers.MaxPooling2D(pool_size=(2, 2)))
```

- `tf.keras.layers.MaxPooling2D`: This adds a max pooling layer to the model.
  - `pool_size=(2, 2)`: The dimensions of the pooling window. Max pooling downsamples the input representation by taking the maximum value over a `2x2` window, reducing the spatial dimensions of the input.

3. **Flatten Layer**

```python
model.add(tf.keras.layers.Flatten())
```

- `tf.keras.layers.Flatten`: This layer flattens the input. It converts the 2D matrix data to a 1D vector, which can be fed into a fully connected (dense) layer. This is necessary before passing the data to dense layers.

Figure 6.17 – Model interpretability with ChatGPT

As you can see in the preceding figure, ChatGPT was able to give us a clear explanation of the structure and layers of our CNN. It also adds some comments and tips, such as the fact that using the max pooling layer helps reduce the dimensionality of the input.

I can also be supported by ChatGPT in interpreting model results in the validation phase. So, after splitting data into training and test sets and training the model on the training set, I want to see its performance on the test set:

```
model.fit(np.resize(x_train, (50000,32,32,1))/255.0,
        tf.keras.utils.to_categorical(y_train),
        batch_size=batch_size,
        shuffle=True,
        epochs=epochs,
        validation_data=(np.resize(x_test, (10000,32,32,1))/255.0,
        tf.keras.utils.to_categorical(y_test))
        )
```

```
Epoch 1/3
1563/1563 [==============================] - 208s 133ms/step - loss: 2.3021 - accuracy: 0.1038 - val_loss: 2.3029 - val_accuracy: 0.1039
Epoch 2/3
1563/1563 [==============================] - 209s 133ms/step - loss: 2.2964 - accuracy: 0.1104 - val_loss: 2.3070 - val_accuracy: 0.1001
Epoch 3/3
1563/1563 [==============================] - 909s 582ms/step - loss: 2.2743 - accuracy: 0.1312 - val_loss: 2.3173 - val_accuracy: 0.1006
```

Figure 6.18 – Evaluation metrics

Let’s also ask ChatGPT to elaborate on our validation metrics (truncated output):

Can you explain to me the output metrics below?

Epoch 1/3
1563/1563 [==========] - 208s 133ms/step - loss: 2.3021
- accuracy: 0.1038 - val_loss: 2.3029 - val_accuracy: 0.1039

Epoch 2/3
1563/1563 [==========] - 209s 133ms/step - loss: 2.2964
- accuracy: 0.1104 - val_loss: 2.3070 - val_accuracy: 0.1001

Epoch 3/3
1563/1563 [==========] - 909s 582ms/step - loss: 2.2743
- accuracy: 0.1312 - val_loss: 2.3173 - val_accuracy: 0.1006

Certainly! Let's break down the output metrics of the training process of a neural network model:

General Format

Each epoch output provides several pieces of information:

1. **Epoch Progress**: Shows the current epoch and total number of epochs.
2. **Batch Progress**: Indicates the completion of batches within the epoch.
3. **Metrics**: Includes training loss, training accuracy, validation loss, and validation accuracy.

Detailed Explanation

Epoch Information

plaintext Copy code

```plaintext
Epoch 1/3
```

- **Epoch 1/3**: This indicates that the training is in the first epoch out of a total of 3 epochs.

Batch Progress and Time

plaintext Copy code

```plaintext
1563/1563 [==========] - 208s 133ms/step
```

- **1563/1563**: Indicates the total number of batches processed in this epoch. Here, all 1563 batches

Figure 6.19 – Example of ChatGPT explaining evaluation metrics

Once again, the result was really impressive, and it provided clear guidance on how to set up ML experiments in terms of training and test sets. It explains how important it is for the model to be sufficiently generalized so that it does not overfit and is able to predict accurate results on data that it has never seen before.

There are many reasons why model interpretability is important. A pivotal element is that it reduces the gap between business users and the code behind models. This is key to enabling business users to understand how a model behaves, as well as translate it into code business ideas.

Furthermore, model interpretability enables one of the key principles of responsible and ethical AI, which is the transparency of how a model behind AI systems thinks and behaves. Unlocking model interpretability means detecting potential biases or harmful behaviors a model could have while in production and consequently preventing them from happening.

Overall, ChatGPT can provide valuable support in the context of model interpretability, generating insights at the row level, as we saw in the previous example.

The next and last ChatGPT capability we will explore will be yet another boost for developers' productivity, especially when various programming languages are being used within the same project.

Translation among different programming languages

In *Chapter 5*, we saw how ChatGPT has great capabilities for translating between different languages. What is really incredible is that natural language is not its only object of translation. In fact, ChatGPT is capable of translating between different programming languages while keeping the same output as well as the same style (namely, it preserves docstring documentation if present).

There are so many scenarios when this could be a game changer.

For example, you might have to learn a new programming language or statistical tool you've never seen before because you need to quickly deliver a project on it. With the help of ChatGPT, you can start programming in your language of preference and then ask it to translate to the desired language, which you will be learning alongside the translation process.

Imagine that the project needs to be delivered in MATLAB (a proprietary numerical computing and programming software developed by MathWorks), yet you've always programmed in Python. The project consists of classifying images from the **Modified National Institute of Standards and Technology** (**MNIST**) dataset (the original dataset description and related paper can be found here at http://yann.lecun.com/exdb/mnist/). The dataset contains numerous handwritten digits and is frequently utilized to teach various image processing systems.

To start, I wrote the following Python code to initialize a deep-learning model for classification:

```
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers
# Load the MNIST dataset
(x_train, y_train), (x_test, y_test) = keras.datasets.mnist.load_ data()
# Preprocess the data
x_train = x_train.reshape(-1, 28*28) / 255.0 x_test = x_test.reshape(-1, 28*28) / 255.0 y_train =
# Define the model architecture model = keras.Sequential([
layers.Dense(256, activation='relu', input_shape=(28*28,)), layers.Dense(128, activation='relu')
layers.Dense(10, activation='softmax')
])
# Compile the model
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
# Train the model
history = model.fit(x_train, y_train, validation_split=0.2, epochs=10, batch_size=128)
# Evaluate the model
test_loss, test_acc = model.evaluate(x_test, y_test, verbose=0) print('Test accuracy:', test_acc
```

Let’s now see what happens if we give the preceding code as context to ChatGPT and ask it to translate it into MATLAB:

Here is the MATLAB translation of the provided TensorFlow code for training a neural network on the MNIST dataset.

MATLAB Code

matlab Copy code

```matlab
% Load the MNIST dataset
[xTrain, yTrain, xTest, yTest] = digitTrain4DArrayData;

% Reshape the data and normalize
xTrain = reshape(xTrain, [], 28*28) / 255.0;
xTest = reshape(xTest, [], 28*28) / 255.0;

% Convert the labels to categorical format
yTrain = categorical(yTrain);
yTest = categorical(yTest);

% Define the model architecture
layers = [
    fullyConnectedLayer(256, 'Activation', 'relu', 'InputSize', 28*28)
    fullyConnectedLayer(128, 'Activation', 'relu')
    fullyConnectedLayer(10, 'Activation', 'softmax')
];

% Create the neural network
model = dlnetwork(layerGraph(layers));

% Specify training options
options = trainingOptions('adam', ...
    'MaxEpochs', 10, ...
    'MiniBatchSize', 128, ...
    'ValidationData', {xTest, yTest}, ...
    'Verbose', true, ...
    'Plots', 'training-progress');

% Train the model
trainedNet = trainNetwork(xTrain, yTrain, model.Layers, options);

% Evaluate the model
YPred = classify(trainedNet, xTest);
testAcc = mean(YPred == yTest);
disp(['Test accuracy:', num2str(testAcc)]);
```

Figure 20 – ChatGPT translates Python code into MATLAB

Let’s also see whether it is capable of translating it into other languages such as JavaScript (note that I didn’t have to repeat the code, since it was already in the chat context):

no translate it into Javascript

Sure! Here's the equivalent code translated into JavaScript using TensorFlow.js.

JavaScript Code

javascript Copy code

```javascript
// Import TensorFlow.js
import * as tf from '@tensorflow/tfjs';
import { MnistData } from './data.js'; // Assuming you have a separate data.js file for lo

async function run() {
    // Load the MNIST dataset
    const data = new MnistData();
    await data.load();

    const [xTrain, yTrain] = data.getTrainData();
    const [xTest, yTest] = data.getTestData();

    // Normalize the data
    const xTrainNorm = xTrain.div(255.0);
    const xTestNorm = xTest.div(255.0);

    // Define the model architecture
    const model = tf.sequential();
    model.add(tf.layers.dense({ units: 256, activation: 'relu', inputShape: [28 * 28] }));
    model.add(tf.layers.dense({ units: 128, activation: 'relu' }));
    model.add(tf.layers.dense({ units: 10, activation: 'softmax' }));

    // Compile the model
    model.compile({
        optimizer: 'adam',
        loss: 'categoricalCrossentropy',
        metrics: ['accuracy']
    });

    // Train the model
    await model.fit(xTrainNorm, yTrain, {
        epochs: 10,
        batchSize: 128,
        validationSplit: 0.2,
        callbacks: tf.callbacks.earlyStopping({ monitor: 'val_loss' })
    });

    // Evaluate the model
    const evalResult = model.evaluate(xTestNorm, yTest);
    const testAcc = evalResult[1].dataSync()[0];
    console.log('Test accuracy:', testAcc);
}

run();
```

Figure 6.21 – ChatGPT translates Python code into JavaScript

Code translation could also reduce the skill gap between new technologies and current programming capabilities.

Another key implication of code translation is **application modernization**. Indeed, imagine you want to refresh your application stack, namely migrating to the cloud. You could decide to initiate with a simple lift and shift going toward **Infrastructure-as-a-Service** (**IaaS**) instances (such as Windows or Linux **virtual machines** (**VMs**)). However, in a second phase, you might want to refactor, rearchitect, or ever rebuild your applications.

The following diagram depicts the various options for application modernization:

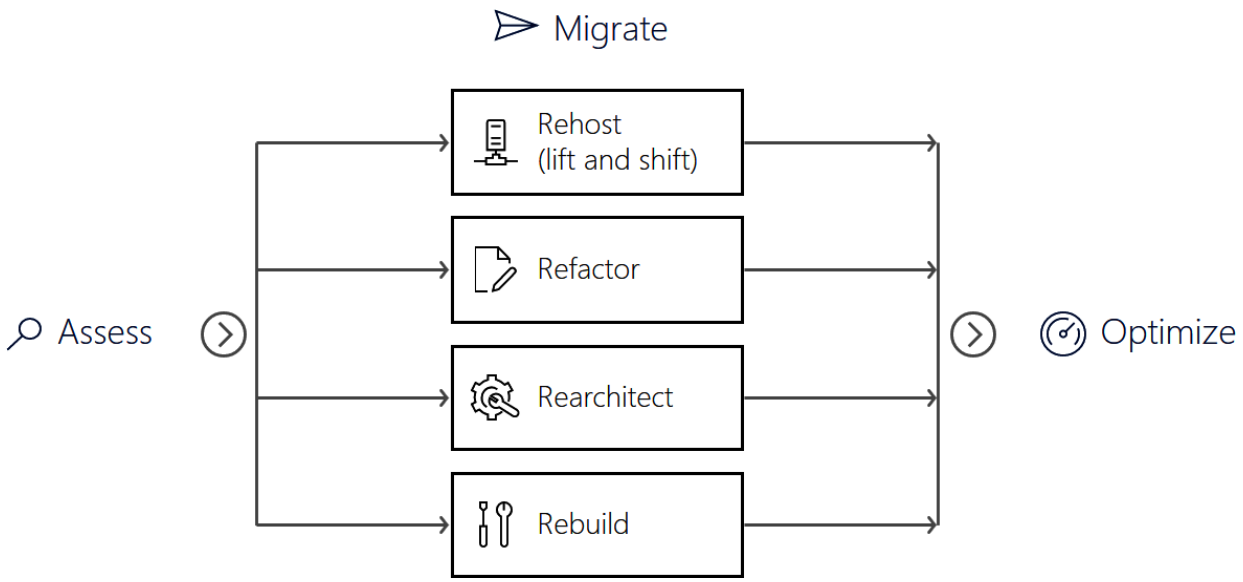


Figure 6.22 – Four ways you can migrate your applications to the public cloud

ChatGPT and OpenAI Codex models can help you with the migration. Consider mainframes, for example.

Mainframes are computers that are predominantly employed by large organizations to carry out essential tasks such as bulk data processing for activities such as censuses, consumer and industry statistics, enterprise resource planning, and large-scale transaction processing. The application programming language of the mainframe environment is **Common Business Oriented Language** (**COBOL**). Despite being invented in 1959, COBOL is still in use today and is one of the oldest programming languages in existence.

As technology continues to improve, applications residing in the realm of mainframes have been subject to a continuous process of migration and modernization aimed at enhancing existing legacy mainframe infrastructure in areas such as interface, code, cost, performance, and maintainability.

Of course, this implies translating COBOL to more modern programming languages such as C# or Java. The problem is that COBOL is unknown to most of the new-generation programmers; hence

there is a huge skills gap in this context.

Let’s consider a COBOL script that reads an input number, adds 10 to it, and then prints the result.

```
IDENTIFICATION DIVISION.
PROGRAM-ID. AddTen.
DATA DIVISION.
WORKING-STORAGE SECTION.
01  INPUT-NUMBER    PIC 9(5).
01  RESULT-NUMBER   PIC 9(5).
PROCEDURE DIVISION.
DISPLAY 'Enter a number: '.
ACCEPT INPUT-NUMBER.
COMPUTE RESULT-NUMBER = INPUT-NUMBER + 10.
DISPLAY 'Result after adding 10: ' RESULT-NUMBER.
STOP RUN.
```

I then passed the previous COBOL script to ChatGPT so that it can use it as context to formulate its response. Let’s now ask ChatGPT to translate that script into C#:

Certainly! Here's the equivalent code translated into C#:

C# Code

csharp Copy code

```csharp
using System;

namespace AddTen
{
    class Program
    {
        static void Main(string[] args)
        {
            // Variable declarations
            int inputNumber;
            int resultNumber;

            // Prompt the user to enter a number
            Console.Write("Enter a number: ");

            // Read input from the user and try to parse it as an integer
            if (int.TryParse(Console.ReadLine(), out inputNumber))
            {
                // Add 10 to the input number
                resultNumber = inputNumber + 10;

                // Display the result
                Console.WriteLine("Result after adding 10: " + resultNumber);
            }
            else
            {
                // Handle invalid input
                Console.WriteLine("Invalid input. Please enter a valid number.");
            }
        }
    }
}
```

Figure 6.23 – Example of ChatGPT translating COBOL to C#.

Tools such as ChatGPT can help in reducing the skill gap in this and similar scenarios by introducing a layer that knows both the past and the future of programming.

In conclusion, ChatGPT can be an effective tool for application modernization, providing code upgrading in addition to valuable insights and recommendations for enhancing legacy systems. With its advanced language processing capabilities and extensive knowledge base, ChatGPT can

help organizations streamline their modernization efforts, making the process faster, more efficient, and more effective.

Summary

ChatGPT can be a valuable resource for developers looking to enhance their skills and streamline their workflows. We started by seeing how ChatGPT can generate, optimize, and debug your code, but we also covered further capabilities such as generating documentation alongside your code, explaining your ML models, and translating between different programming languages for application modernization.

Whether you're a seasoned developer or just starting out, ChatGPT offers a powerful tool for learning and growth, reducing the gap between code and natural language.

In the next chapter, we will dive deeper into another domain of application where ChatGPT could be a game changer: marketing.

6 Mastering Marketing with ChatGPT

Join our book community on Discord

https://packt.link/EarlyAccess

In this chapter, we will focus on how marketers can leverage ChatGPT, looking at the main use cases of ChatGPT in this domain, and how marketers can leverage it as a valuable assistant.

We will learn how ChatGPT can assist in the following activities:

- Marketers' need for ChatGPT
- New product development and the go-to-market strategy
- A/B testing for marketing comparison
- Making more efficient websites and posts with **Search Engine Optimization** (**SEO**)
- Sentiment analysis of textual data

By the end of this chapter, you will be able to leverage ChatGPT for marketing-related activities and to boost your productivity.

Technical requirements

You will need an OpenAI account to access ChatGPT.

Marketers' need for ChatGPT

Marketing is probably the domain where ChatGPT and OpenAI models' creative power can be leveraged in their purest form. They can be practical tools to support creative development in terms of new products, marketing campaigns, search engine optimization, and so on. Overall, marketers automate and streamline many aspects of their work, while also improving the quality and effectiveness of their marketing efforts.

Here is an example. One of the most prominent and promising use cases of ChatGPT in marketing is personalized marketing. ChatGPT can be used to analyze customer data and generate personalized marketing messages that resonate with individual customers. For example, a marketing team can use ChatGPT to analyze customer data and develop targeted email campaigns that are tailored to specific customer preferences and behavior. This can increase the likelihood of conversion and lead to greater customer satisfaction. By providing insights into customer sentiment and behavior, generating personalized marketing messages, providing personalized customer support, and generating content, ChatGPT can help marketers deliver exceptional customer experiences and drive business growth.

This is one of many examples of ChatGPT applications in marketing. In the following sections, we will look at concrete examples of end-to-end marketing projects supported by ChatGPT.

New product development and the go-to-market strategy

The first way you can introduce ChatGPT into your marketing activity might be as an assistant in new product development and **go-to-market** (**GTM**) strategy.

In this section, we will look at a step-by-step guide on how to develop and promote a new product. You already own a running clothing brand called **RunFast** and so far you have only produced shoes, so you want to expand your business with a new product line. We will start by brainstorming ideas to create a GTM strategy. Of course, everything is supported by ChatGPT:

- Brainstorming ideas: The first thing ChatGPT can support you with is brainstorming and drafting options for your new product line. It will also provide the reasoning behind each suggestion. So, let’s ask what kind of new product line I should focus on:

I own a running clothing brand that produces running shoes. Generate 3 new product ideas I can add as product lines for my brand

Memory updated

Here are three new product ideas you can add to your running clothing brand:

1. **Smart Running Apparel**:
   - **Description**: High-tech running clothes equipped with sensors to monitor heart rate, steps, and calories burned. The apparel could include shirts, shorts, and socks with integrated technology that syncs with a mobile app to provide real-time feedback and performance tracking.
   - **Key Features**: Breathable and moisture-wicking fabric, seamless integration with fitness apps, washable electronics, and customizable alerts for hydration and pace.
2. **Eco-Friendly Running Gear**:
   - **Description**: A line of environmentally sustainable running gear made from recycled materials. This could include jackets, leggings, and hats designed for different weather conditions, promoting eco-conscious running.
   - **Key Features**: Recycled polyester fabric, biodegradable packaging, durable and high-performance material, and a program for recycling old running gear.
3. **Personalized Running Shoes**:
   - **Description**: Customizable running shoes tailored to the individual's foot shape, running style, and personal preferences. Using advanced 3D printing technology and a user-friendly online design tool, runners can create shoes that fit perfectly and reflect their style.
   - **Key Features**: 3D-printed insoles for perfect fit, a variety of design options (colors, patterns, logos), materials chosen based on running needs (trail, road, marathon), and virtual fitting sessions with experts.

Figure 7.1 – Example of new ideas generated by ChatGPT

Out of the three suggestions, we will pick the second one, because of the positive environmental impact that we can make with it, as well as improving our brand reputation. More specifically, I will start with eco-friendly running socks.

- Product name: Now that we have our idea fixed in mind, we need to think of a catchy name for it. Again, I will ask ChatGPT for more options so that I can then pick my favorite one:

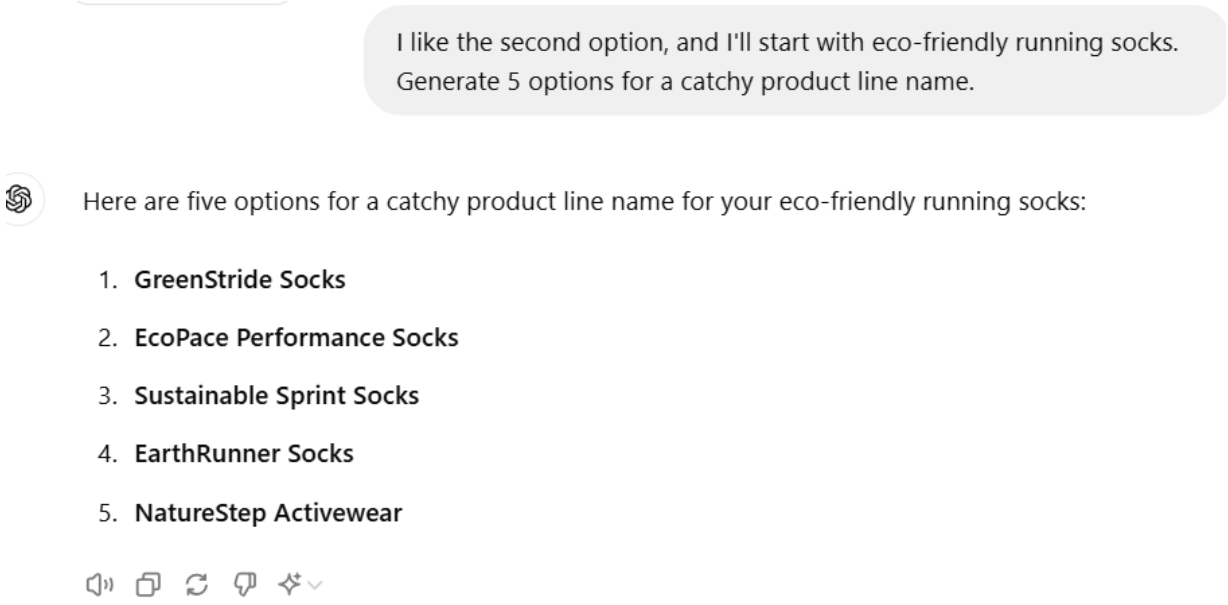


Figure 7.2 – A list of potential product names

GreenStride sounds good enough for me – I'll go ahead with that one.

- Generating catchy slogans: On top of the product name, I also want to share the intent behind the name and the mission of the product line, so that my target audience is captured by it. I want to inspire trust and loyalty in my customers and for them to see themselves reflected in the mission behind my new product line.

Generate 3 slogans for the "GreenStride" name. They should be motivating and concise.

Here are three motivating and concise slogans for the "GreenStride" name:

1. **"Run Green, Run Strong."**
2. **"Step Forward, Leave No Trace."**
3. **"Stride with Purpose."**

Figure 7.3 – A list of slogans for my new product name

Great – now I'm satisfied with the product name and slogan that I will use later on to create a unique social media announcement. Before doing that, I want to spend more time on market research for the target audience.

what kind of target audience should I address with the promotion of GreenStride socks product line?

For promoting the "GreenStride" socks product line, you should target the following audience segments:

1. **Eco-Conscious Runners**:
   - **Description**: Individuals who prioritize sustainability and environmentally friendly products. They are willing to invest in eco-friendly alternatives and are often knowledgeable about environmental issues.
   - **Demographics**: Ages 25-45, urban dwellers, middle to high income, active on social media, particularly platforms focused on sustainability and fitness.
   - **Marketing Channels**: Social media campaigns, environmental blogs, partnerships with eco-friendly influencers, and sustainability forums.
2. **Fitness Enthusiasts**:
   - **Description**: People who are passionate about running and maintaining an active lifestyle. They value high-performance gear and are interested in the latest trends and innovations in running apparel.
   - **Demographics**: Ages 20-40, both genders, middle to high income, frequent participants in marathons, races, and running clubs.
   - **Marketing Channels**: Running and fitness magazines, sponsorships at running events, online fitness communities, and targeted ads on fitness apps.
3. **Health-Conscious Consumers**:
   - **Description**: Individuals focused on overall wellness, including physical activity, healthy eating, and sustainable living. They are looking for products that support their healthy lifestyle and ethical values.
   - **Demographics**: Ages 30-50, both genders, middle to high income, likely to participate in yoga, pilates, and other wellness activities.
   - **Marketing Channels**: Health and wellness blogs, collaborations with wellness influencers, email newsletters, and health-focused retail stores.

Figure 7.4 – List of groups of target people to reach with my new product line

It’s important to have in mind different clusters within your audience so that you can differentiate the messages you want to give. In my case, I want to make sure that my product line will address different groups of people, such as competitive runners, casual runners, and fitness enthusiasts.

- Product variants and sales channels: According to the preceding clusters of potential customers, I could generate product variants so that they are more tailored toward specific audiences:

regarding the two segments of Eco-conscious runners and Health-conscious consumers, suggest me socks product variants that are specifically tailored for those segments.

Eco-Conscious Runners

1. **Recycled Fiber Performance Socks**:
   - **Description**: Socks made from 100% recycled materials, including plastic bottles and discarded fabrics. These socks provide excellent moisture-wicking and breathability for long runs.
   - **Key Features**: High breathability, moisture-wicking, and anti-odor properties, reinforced heel and toe for durability, seamless design for comfort.
2. **Organic Cotton Running Socks**:
   - **Description**: Soft and comfortable running socks made from organic cotton, grown without synthetic pesticides or fertilizers.
   - **Key Features**: Hypoallergenic, ultra-soft texture, natural moisture absorption, and eco-friendly dyes.
3. **Bamboo Fiber Eco Socks**:
   - **Description**: Socks made from sustainable bamboo fibers, offering natural antimicrobial and moisture-wicking properties.
   - **Key Features**: Superior softness, natural odor resistance, excellent breathability, and reduced environmental impact.

Health-Conscious Consumers

1. **Compression Running Socks**:
   - **Description**: Graduated compression socks designed to improve blood circulation and reduce muscle fatigue during and after running.
   - **Key Features**: Enhanced circulation, reduced muscle fatigue, quicker recovery times, and snug fit for added support.
2. **Cushioned Support Socks**:
   - **Description**: Extra-cushioned running socks providing additional support and shock absorption, ideal for individuals focused on joint and muscle health.
   - **Key Features**: Thick cushioning, arch support, reduced impact on joints, and plush comfort for long-distance runs.
3. **Anti-Blister Running Socks**:
   - **Description**: High-performance socks designed to prevent blisters, featuring a double-layered design to minimize friction and maximize comfort.
   - **Key Features**: Double-layered fabric, smooth seams, moisture-wicking, and anti-friction properties.

By offering these product variants, you can cater to the specific needs and preferences of eco-conscious runners and health-conscious consumers, ensuring that the "GreenStride" socks product line meets their expectations for sustainability and performance.

Figure 7.5 – Example of variants of the product line

Similarly, I can also ask ChatGPT to suggest different sales channels for each of the preceding groups:

what could be the best channel to reach each of the two segments above?

Best Channels to Reach Eco-Conscious Runners

1. **Social Media Campaigns**:
   - **Platforms**: Instagram, Facebook, and Pinterest.
   - **Strategies**: Use visually appealing content that highlights the eco-friendly aspects of the socks, collaborate with eco-friendly influencers, and participate in environmental awareness campaigns.
2. **Environmental Blogs and Websites**:
   - **Platforms**: Popular eco-conscious blogs, sustainability-focused websites.
   - **Strategies**: Write guest posts, engage in sponsored content, and provide informative articles about the sustainable practices used in the production of GreenStride socks.
3. **Partnerships with Eco-Friendly Organizations**:
   - **Organizations**: Environmental NGOs, sustainability-focused running clubs, and eco-conscious events.
   - **Strategies**: Sponsor events, provide product samples for eco-marathons, and create joint marketing campaigns with these organizations.
4. **Email Newsletters**:
   - **Platforms**: Email marketing services like Mailchimp or Constant Contact.
   - **Strategies**: Create informative newsletters that highlight the environmental benefits of GreenStride socks, share stories about sustainable practices, and offer special promotions

Best Channels to Reach Health-Conscious Consumers

1. **Health and Wellness Blogs and Magazines**:
   - **Platforms**: Popular health blogs, online wellness magazines, and physical fitness magazines.
   - **Strategies**: Contribute articles, engage in product reviews, and offer sponsored content focused on the health benefits of wearing GreenStride socks.
2. **Fitness and Wellness Influencers**:
   - **Platforms**: Instagram, YouTube, and TikTok.
   - **Strategies**: Collaborate with fitness influencers to create content that showcases the health benefits of the socks, such as improved circulation and muscle support.
3. **Online Fitness Communities and Forums**:
   - **Platforms**: Reddit, fitness-related forums, and health-focused online communities.
   - **Strategies**: Engage in discussions, share success stories and testimonials, and provide expert advice on the benefits of using GreenStride socks for running and overall wellness.
4. **Health-Focused Retail Stores**:
   - **Stores**: Specialty health stores, fitness gear shops, and wellness centers.
   - **Strategies**: Partner with these stores to carry GreenStride socks, offer in-store promotions, and provide educational material about the health benefits of the socks.

By leveraging these targeted channels, you can effectively reach and engage with both eco-conscious runners and health-conscious consumers, driving awareness and sales for your GreenStride socks product line.

Figure 7.6 – Suggestions for different sales channels by ChatGPT

- Standing out from the competition: I want my product line to stand out from the competition and emerge in a very saturated market – I want to make it unique. With this purpose in mind, I asked ChatGPT to include social considerations such as sustainability and inclusivity. Let's ask ChatGPT for some suggestions in that respect:

give me three concise suggestions on how to make my socks line GreenStride outstading and unique in a competitive market.

Here are three concise suggestions to make your GreenStride socks line outstanding and unique in a competitive market:

1. **Innovative Eco-Materials**:
   - Use cutting-edge, sustainable materials like recycled ocean plastics, organic hemp, or biodegradable fibers. Highlight these unique materials in your marketing to emphasize the advanced sustainability of your product.
2. **Performance-Enhancing Features**:
   - Integrate advanced features such as targeted compression zones, moisture-wicking technology, and anti-blister designs. Ensure these features are scientifically validated and clearly communicated to consumers to showcase the dual focus on sustainability and performance.
3. **Transparent Production Process**:
   - Offer full transparency about your production process, from sourcing to manufacturing. Provide detailed information and certifications about the eco-friendly practices and fair labor conditions involved in making GreenStride socks. Transparency builds trust and can be a significant differentiator.

Figure 7.7 – Example of outstanding features generated by ChatGPT

As you can see, it was able to generate interesting features that could make my product line unique.

- Product description: Now it's time to start building our Go-To-Market plan. First of all, I want to generate a product description for my website, including all the earlier unique differentiators.

generate a product description (max 150 words) for GreenStride socks line using the unique differentiator you listed above. It should be grabbing and effective, as well as SEO optimized. list also the SEO keywords you used once you finish.

GreenStride Socks Product Description

Step into the future of running with GreenStride socks, crafted from innovative eco-materials like recycled ocean plastics and organic hemp. Designed for superior performance, our socks feature targeted compression zones, moisture-wicking technology, and anti-blister designs to keep you comfortable and focused on your run. With GreenStride, you're not just choosing top-notch running gear—you're supporting a sustainable future. We're committed to transparency, offering full insight into our eco-friendly production process and fair labor practices. Elevate your stride with GreenStride, where sustainability meets performance.

SEO Keywords

- Eco-friendly running socks
- Sustainable performance socks
- Recycled materials socks
- Organic hemp socks
- Anti-blister running socks
- Moisture-wicking technology
- Targeted compression zones
- Transparent production process
- Sustainable athletic wear

Figure 7.8 – Example of description and SEO keywords generated by ChatGPT

- Fair price: Another key element is determining a fair price for our product. As I differentiated among product variants for different audiences (competitive runners, casual runners, and fitness enthusiasts), I also want to have a price range that takes into account this clustering. Note that, in the following example, ChatGPT is invoking the web search plug-in to retrieve updated information about the current marketplace of running socks in terms of pricing.

what could be a fair price for my socks line, considering the two audience segments above and a current analysis of the running socks market?

Searched 6 sites

Based on the current market analysis, a fair price for your "GreenStride" socks line can be determined by considering the pricing trends for similar high-quality and eco-friendly athletic socks.

Market Insights:

1. **Price Segments:**
   - **Low-End Socks:** Generally priced around $5-$10 per pair. These are often made from basic materials without advanced features.
   - **Medium-Range Socks:** Priced between $10-$20 per pair. This segment balances quality and affordability, featuring better materials and some advanced features.
   - **High-End Socks:** Typically priced above $20 per pair. These include premium materials, advanced technologies, and sustainability certifications.
2. **Eco-Friendly and Performance Socks:**
   - The market for sustainable and performance-oriented socks is growing, with consumers willing to pay a premium for high-quality, eco-friendly products. Brands like Nike and Adidas offer high-performance athletic socks in this range, often priced between $15-$25 per pair (Mordor Intelligence) (Grand View Research) (Market Data Forecast).

Recommendations for "GreenStride" Socks:

Considering your target segments of eco-conscious runners and health-conscious consumers, positioning your socks in the **medium to high price range** would be appropriate. This range reflects the sustainable materials and advanced performance features your socks offer.

Suggested Pricing:

- **Standard Eco-Friendly Performance Socks:** $15-$18 per pair.
- **Premium Features (e.g., extra cushioning, advanced moisture-wicking):** $20-$25 per pair.

This pricing strategy aligns with market expectations for high-quality, sustainable athletic wear and ensures competitiveness while highlighting the unique value propositions of the "GreenStride" line (MAXIMIZE MARKET RESEARCH) (Global Market Insights Inc.).

Figure 7.9 – Price ranges for product variants

We are almost there. We have gone through many new product development and go-to-market steps, and in each of them, ChatGPT acted as a great support tool.

As one last thing, we can ask ChatGPT to generate an Instagram post about our new product, including relevant hashtags and SEO keywords. We can then generate the image DALL-E, which comes as an embedded plug-in in ChatGPT Plus.

I want to generate an Instagram post to announce the launch of GreenStride socks. Write a post (max 150 words) including the unique features and differentiators mentioned above, as well as relevant hashtags.

Instagram Post for GreenStride Socks Launch

🌿 **Introducing GreenStride Socks!** 🌿

Step into a new era of running with GreenStride, where sustainability meets top-tier performance. Crafted from innovative eco-materials like recycled ocean plastics and organic hemp, our socks offer targeted compression, advanced moisture-wicking technology, and anti-blister designs. Each pair is made with the environment in mind, ensuring you can run green and run strong. 🏋️‍♀️💪

Join the movement towards a sustainable future with socks that don't just perform—they make a difference. 🌎✨

🧦 **Key Features:**

- Recycled ocean plastics & organic hemp
- Targeted compression zones
- Superior moisture-wicking
- Anti-blister design

Get ready to elevate your stride and make every step count! 🏃‍♀️🌿

#GreenStride #EcoFriendly #SustainableRunning #RunGreen #EcoConscious #AthleticWear #RunStrong #PerformanceSocks #MoistureWicking #AntiBlister #HealthyLiving #SustainableFashion

Figure 7.10 – Social media post generated by ChatGPT

And, with the special contribution of DALL-E:

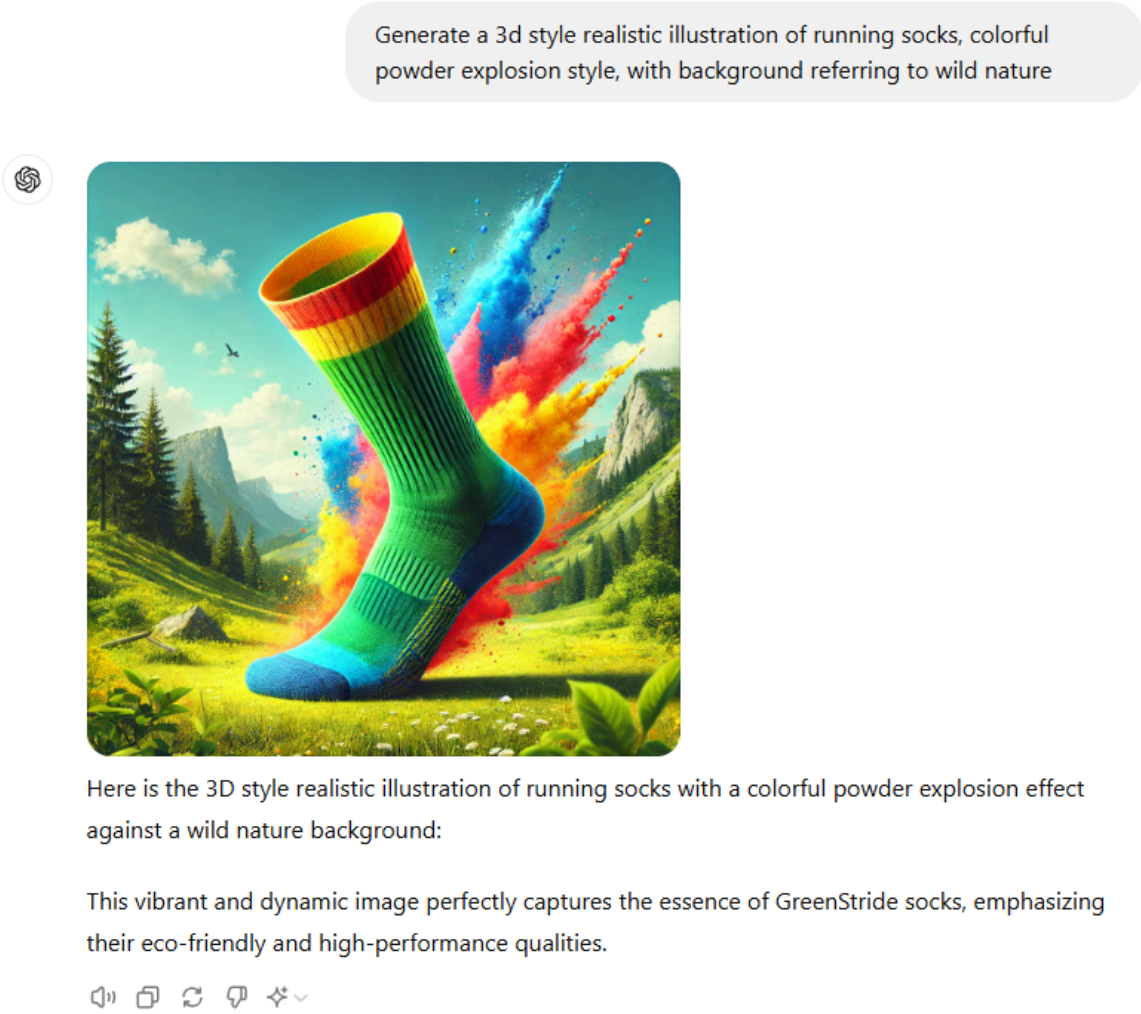


Figure 1: Example of an illustration generated by ChatGPT powered by DALL-E 3.

Here it is the final result:

Instagram

🌿 **Introducing GreenStride Socks!** 🌿
Step into a new era of running with GreenStride, where sustainability meets top-tier performance. Crafted from innovative eco-materials like recycled ocean plastics and organic hemp, our socks offer targeted compression, advanced moisture-wicking technology, and anti-blister designs. Each pair is made with the environment in mind, ensuring you can run green and run strong. ⏳ 💪
Join the movement towards a sustainable future with socks that don't just perform—they make a difference. 🌎 ✨
🧦 **Key Features**:
•Recycled ocean plastics & organic hemp
•Targeted compression zones
•Superior moisture-wicking
•Anti-blister design
Get ready to elevate your stride and make every step count! 🦶 🌿
#GreenStride #EcoFriendly #SustainableRunning #RunGreen #EcoConscious #AthleticWear #RunStrong #PerformanceSocks #MoistureWicking #AntiBlister

#HealthyLiving #SustainableFashion

Figure 7.11 – Instagram post entirely generated by ChatGPT and DALL-E

Of course, many elements are missing here for complete product development and go-to-market. Yet, with the support of ChatGPT (and the special contribution of DALL-E 3) we managed to brainstorm a new product line and variants, potential customers, catchy slogans, and finally, generated a pretty nice Instagram post to announce the launch of GreenStride!

A/B testing for marketing comparison

Another interesting field where ChatGPT can assist marketers is A/B testing.

A/B testing in marketing is a method of comparing two different versions of a marketing campaign, advertisement, or website to determine which one performs better. In A/B testing, two variations of the same campaign or element are created, with only one variable changed between the two versions. The goal is to see which version generates more clicks, conversions, or other desired outcomes.

An example of A/B testing might be testing two versions of an email campaign, using different subject lines, or testing two versions of a website landing page, with different call-to-action buttons. By measuring the response rate of each version, marketers can determine which version performs better and make data-driven decisions about which version to use going forward.

A/B testing allows marketers to optimize their campaigns and elements for maximum effectiveness, leading to better results and a higher return on investment.

Since this method involves the process of generating many variations of the same content, the generative power of ChatGPT can definitely assist in that.

Let’s consider the following example. I’m promoting a new product I developed: a new, light and thin climbing harness for speed climbers. I’ve already done some market research and I know my niche audience. I also know that one great channel of communication for that audience is publishing on an online climbing blog, of which most climbing gyms’ members are fellow readers.

My goal is to create an outstanding blog post to share the launch of this new harness, and I want to test two different versions of it in two groups. The blog post I’m about to publish and that I want to be the object of my A/B testing is the following:

Lighter Climbing Harness for Speed Climbers
Introducing the Latest Addition to the World of Climbing Gear

Attention climbers! Upgrade your gear with the latest harness designed for speed climbers. It's lighter than any other, allowing you to move faster and more efficiently.

This harness improves speed and performance without sacrificing safety. Made from lightweight, breathable, and durable material, it's perfect for both beginners and seasoned climbers. Head to your nearest gear store and elevate your climbing experience today!

Figure 7.12 – An example of a blog post to launch climbing gear

Here, ChatGPT can help us on two levels:

- The first level is that of rewording the article, using different keywords or different attention- grabbing slogans. To do so, once this post is provided as context, we can ask ChatGPT to work on the article and slightly change some elements:

I'm doing an A/B testing for the launch of a new climbing harness. I'm using a climbing blog as sales channel with the below blog post. Please regenerate the below content changing the following elements: title, subtitle and closing sentence.

Lighter Climbing Harness for Speed Climbers
Introducing the Latest Addition to the World of Climbing Gear

Attention climbers! Upgrade your gear with the latest harness designed for speed climbers. It's lighter than any other, allowing you to move faster and more efficiently.

This harness improves speed and performance without sacrificing safety. Made from lightweight, breathable, and durable material, it's perfect for both beginners and seasoned climbers. Head to your nearest gear store and elevate your climbing experience today!

Elevate Your Climbing Game with Our New Harness

The Ultimate Gear for Speed and Efficiency

Hey climbers! Upgrade to our newest harness specifically designed for speed climbing. Experience unmatched agility and efficiency with our lightest harness yet.

Engineered to enhance your speed and performance while ensuring maximum safety, this harness is crafted from lightweight, breathable, and durable materials. Ideal for both novices and expert climbers. Visit your favorite gear store and take your climbing to the next level today!

Figure 7.13 – New version of the blog post generated by ChatGPT

As per my request, ChatGPT was able to regenerate only those elements I asked for (title, subtitle, and closing sentence) so that I can monitor the effectiveness of those elements by monitoring the reaction of the two audience groups.

- The second level is working on the design of the web page, namely, changing the collocation of the image rather than the position of the buttons. For this purpose, I created a

simple web page for the blog post published in the climbing blog (you can find the code in the book's GitHub repository at https://github.com/PacktPublishing/The-Ultimate-Guide-to-ChatGPT-and-OpenAI/tree/main/Chapter%207%20-%20ChatGPT%20 for%20Marketers/Code):

Figure 7.14 – Sample blog post published on the climbing blog

We can directly feed ChatGPT with the HTML code and ask it to change some layout elements, such as the position of the buttons or their wording. For example, rather than Buy Now, a reader might be more gripped by an I want one! button.

So, lets feed ChatGPT with the HTML source code:

Please regenerate the following HTML code for the three buttons so that, instead of "Buy Now", the text says "I want one!". Also, make buttons more rounded and all of the same color.

```
<!-- Buy Button -->
<a href="http://example.com/buy-now">Buy Now</a>

<!-- Share Button -->
<a href="https://www.facebook.com/sharer/sharer.php?u=http://example.com/your-article" target="_blank" class="facebook">Share on Facebook</a>

<!-- Feedback Button -->
<a href="http://example.com/feedback">Leave Feedback</a>
```

Here is the regenerated HTML code for the buttons with the updated text, rounded corners, and consistent color:

html Copy code

```html
<!-- Buy Button -->
<a href="http://example.com/buy-now" style="display:inline-block; padding:10px 20px; color

<!-- Share Button -->
<a href="https://www.facebook.com/sharer/sharer.php?u=http://example.com/your-article" tar

<!-- Feedback Button -->
<a href="http://example.com/feedback" style="display:inline-block; padding:10px 20px; colo
```

All buttons now have the text "I want one!" (for the buy button), rounded corners with a border-radius of 25px, and the same background color (#4CAF50).

Figure 7.14 – ChatGPT changing HTML code

Let's see what the output looks like (I also changed the title, subtitle and paragraph with the once generated by ChatGPT):

Elevate Your Climbing Game with Our New Harness

The Ultimate Gear for Speed and Efficiency

Hey climbers! Upgrade to our newest harness specifically designed for speed climbing. Experience unmatched agility and efficiency with our lightest harness yet.

Engineered to enhance your speed and performance while ensuring maximum safety, this harness is crafted from lightweight, breathable, and durable materials.

Ideal for both novices and expert climbers. Visit your favorite gear store and take your climbing to the next level today!

I want one! | Share on Facebook | Leave Feedback

Figure 7.15 – New version of the website

As you can see, ChatGPT only intervened at the button level, slightly changing their layout, position, color, and wording.

In conclusion, ChatGPT is a valuable tool for A/B testing in marketing. Its ability to quickly generate different versions of the same content can reduce the time to market of new campaigns. By utilizing ChatGPT for A/B testing, you can optimize your marketing strategies and ultimately drive better results for your business.

Boosting Search Engine Optimization (SEO)

Another promising area for ChatGPT to be a game changer is **Search Engine Optimization** (**SEO**). This is the key element behind ranking in search engines such as Google or Bing and it determines whether your websites will be visible to users who are looking for what you promote.

> SEO is a technique used to enhance the visibility and ranking of a website on **search engine results pages** (**SERPs**). It is done by optimizing the website or web page to increase the amount and quality of organic (unpaid) traffic from search engines. The purpose of SEO is to attract more targeted visitors to the website by optimizing it for specific keywords or phrases.

Imagine you run an e-commerce company called Hat&Gloves, which only sells, as you might have guessed, hats and gloves. You are now creating your e-commerce website and want to

optimize its ranking. Let’s ask ChatGPT to list some relevant keywords to embed in our website:

I'm building a website for my e-commerce called "Hat&Gloves", that only sells hats and gloves. I want to optimize the ranking of my websites. Please list 8 keywords I should embed in my website code in order to have SEO relevant results

To optimize the ranking of your "Hat&Gloves" e-commerce website, you should focus on embedding keywords that are both relevant to your products and commonly searched by potential customers. Here are eight effective keywords:

1. **hats**
2. **gloves**
3. **winter hats**
4. **fashion gloves**
5. **men's hats**
6. **women's gloves**
7. **beanies**
8. **leather gloves**

These keywords are specific and targeted, which can help improve your website's search engine optimization (SEO) by aligning with common search queries related to your products.

Figure 7.17 – Example of SEO keywords generated by ChatGPT

As you can see, ChatGPT was able to create a list of keywords of different kinds. Some of them are pretty intuitive, such as **Hats** and **Gloves**. Others are related, with an indirect link. For example, **Gift ideas** is not necessarily related to my e-commerce business, however, it could be very smart to include it in my keywords, so that I can widen my audience.

Another key element of SEO is **search engine intent**. Search engine intent, also known as **user intent**, refers to the underlying purpose or goal of a specific search query made by a user in a search engine. Understanding search engine intent is important because it helps businesses and marketers create more targeted and effective content and marketing strategies that align with the searcher's needs and expectations.

There are generally four types of search engine intent:

- **Informational intent**: The user is looking for information on a particular topic or question, such as What is the capital of France? or How to make a pizza at home.
- **Navigational intent**: The user is looking for a specific website or web page, such as **Facebook login** or **Amazon.com**.
- **Commercial intent**: The user is looking to buy a product or service, but may not have made a final decision yet. Examples of commercial intent searches include *best laptop under $1000* or *discount shoes online*.
- **Transactional intent**: The user has a specific goal to complete a transaction, which might refer to physical purchases or subscribing to services. Examples of transactional intent could be *buy iPhone 13* or *sign up for a gym membership*.

By understanding the intent behind specific search queries, businesses and marketers can create more targeted and effective content that meets the needs and expectations of their target audience. This can lead to higher search engine rankings, more traffic, and ultimately, more conversions and revenue.

Now, the question is, will ChatGPT be able to determine the intent of a given request? Before answering, it is worth noticing that the activity of inferring the intent of a given prompt is the core business of **Large Language Models** (**LLMs**), including GPT. So, for sure, ChatGPT is able to capture prompts' intents.

The added value here is that we want to see whether ChatGPT is able to determine the intent in a precise domain with a precise taxonomy, that is, the one of marketing. That is the reason why prompt design is once again pivotal in guiding ChatGPT in the right direction.

Organize the above keyword list in groups based on their search intent, whether commercial, transactional, navigational or informational

Here is the keyword list organized by their search intent:

Commercial Intent

1. **fashion gloves**
2. **leather gloves**

Transactional Intent

1. **hats**
2. **gloves**
3. **winter hats**
4. **men's hats**
5. **women's gloves**
6. **beanies**

Navigational Intent

(None of the provided keywords fit this category as they are more focused on product-specific searches.)

Informational Intent

(None of the provided keywords fit this category as they are more focused on product-specific searches.)

These keywords reflect the intent of users looking to either purchase specific types of hats and gloves (transactional), or explore fashion and material options (commercial). This organization helps in understanding and targeting the right audience more effectively for SEO purposes.

Figure 7.18 – Example of keywords clustered by user intent by ChatGPT

Finally, we could also go further and leverage once more the Act as… hack, which we already mentioned in *Chapter 3*. It would be very interesting indeed to have an assessment of our website and understand whether it is optimized as intended. In

marketing, this analysis is called an **SEO audit**. An SEO audit is an evaluation of a website's SEO performance and potential areas for improvement. An SEO audit is typically conducted by SEO experts, web developers, or marketers, and involves a comprehensive analysis of a website's technical infrastructure, content, and backlink profile.

During an SEO audit, the auditor will typically use a range of tools and techniques to identify areas of improvement, such as keyword analysis, website speed analysis, website architecture analysis, and content analysis. The auditor will then generate a report outlining the key issues, opportunities for improvement, and recommended actions to address them.

Let's ask ChatGPT to act as an SEO expert to conduct this audit. As a reference website, we will use our climbing blog referenced above. I will give ChatGPT the code and ask give the following instructions: "act as a SEO specialist and generate a brief SEO audit (max 300 words) on the above HTML code". This is the response:

SEO Audit for "Lighter Climbing Harness for Speed Climbers" Webpage

Title and Meta Description

- **Title**: The title "Lighter Climbing Harness for Speed Climbers" is concise and contains relevant keywords. However, adding more context or a unique selling proposition could enhance click-through rates. For example, "Lighter Climbing Harness for Speed Climbers - Enhance Performance & Safety".
- **Meta Description**: A meta description is missing. Adding a meta description with keywords can improve search engine visibility. Example: "Discover our new lightweight climbing harness designed for speed climbers. Move faster and more efficiently with our durable, breathable gear."

Content Optimization

- **Headers**: The use of headers (h1, h2) is good, but incorporating keywords into the headers could improve SEO. For instance, "Introducing the Latest Addition to Climbing Gear - Lightweight Harness for Speed Climbers".
- **Keywords**: The content includes relevant keywords like "climbing harness" and "speed climbers". Additional keywords such as "lightweight harness", "climbing gear", and "performance climbing" should be naturally integrated.

Image Optimization

- **Alt Text**: The image alt text "Climbing harness" is appropriate but could be more descriptive. Example: "Lightweight climbing harness for speed climbers".
- **File Name**: Ensure the image file name is descriptive and includes keywords, such as "lightweight-climbing-harness.png".

Internal Linking

- **Links**: The page lacks internal links. Including links to related products or blog posts can improve SEO. Example: "Learn more about our climbing gear".

URL Structure

- **URL**: The URLs for the buttons are placeholders. Ensure the final URLs are clean and include keywords, e.g., "http://example.com/buy-climbing-harness".

Mobile Optimization

- **Responsive Design**: The website uses a responsive design, which is good for mobile SEO. Ensure all elements are mobile-friendly.

Call to Action (CTA)

- **Buttons**: The CTAs are clear, but consider adding more descriptive text for SEO. Example: "Buy Now - Lightweight Climbing Harness" instead of just "Buy Now".

Load Time

- **Performance**: Check the website's load time and optimize images and code to improve speed, as faster load times improve user experience and SEO.

SEO Keywords:

1. **lightweight climbing harness**
2. **speed climbers gear**
3. **climbing gear**
4. **performance climbing harness**
5. **breathable climbing harness**
6. **durable climbing harness**
7. **best climbing harness**
8. **climbing safety gear**

Conclusion

Enhancing the meta tags, integrating more relevant keywords, optimizing images, adding internal links, and ensuring mobile optimization will significantly improve the SEO performance of this webpage.

Figure 7.19 – ChatGPT generating an SEO audit on the HTML code of a climbing blog.

ChatGPT was able to generate a pretty accurate analysis, with relevant comments and suggestions. Overall, ChatGPT has interesting potential for SEO-related activities, and it can be a good tool whether you are building your website from scratch, or you want to improve existing ones.

Sentiment analysis to improve quality and increase customer satisfaction

Sentiment analysis is a technique used in marketing to analyze and interpret the emotions and opinions expressed by customers toward a brand, product, or service. It involves the use of **natural language processing** (**NLP**) and **machine learning** (**ML**) algorithms to identify and classify the sentiment

of textual data such as social media posts, customer reviews, and feedback surveys.

By performing sentiment analysis, marketers can gain insights into customer perceptions of their brand, identify areas for improvement, and make data-driven decisions to optimize their marketing strategies. For example, they can track the sentiment of customer reviews to identify which products or services are receiving positive or negative feedback and adjust their marketing messaging accordingly.

Overall, sentiment analysis is a valuable tool for marketers to understand customer sentiment, gauge customer satisfaction, and develop effective marketing campaigns that resonate with their target audience.

Sentiment analysis has been around for a while, so you might be wondering what ChatGPT could bring as added value. Well, besides the accuracy of the analysis (it being the most powerful model on the market right now), ChatGPT differentiates itself from other sentiment analysis tools since it is **general artificial intelligence** (**AGI**).

This means that when we use ChatGPT for sentiment analysis, we are not using one of its specific APIs for that task: the core idea behind ChatGPT and OpenAI models is that they can assist the user in many general tasks at once, interacting with a task and changing the scope of the analysis according to the user's request.

So, for sure, ChatGPT is able to capture the sentiment of a given text, such as a Twitter post or a product review. However, ChatGPT can also go further and assist in identifying specific

aspects of a product or brand that are positively or negatively impacting the sentiment. For example, if customers consistently mention a particular feature of a product in a negative way, ChatGPT can highlight that feature as an area for improvement. Or, ChatGPT might be asked to generate a response to a particularly delicate review, keeping in mind the sentiment of the review and using it as context for the response. Again, it can generate reports that summarize all the negative and positive elements found in reviews or comments and cluster them into categories.

Let’s consider the following example. A customer has recently purchased a pair of shoes from my e-commerce company, RunFast, and left the following review:

“I recently bought the RunFast Prodigy shoes and have mixed feelings. They're extremely comfortable with excellent cushioning and support, reducing foot fatigue during my runs. The design is also appealing, and I've received several compliments. However, the durability is disappointing; the outsole wears quickly, and the breathable upper shows signs of wear after a few weeks. Given the high price, I'm hesitant to recommend them despite their comfort and design.”

Let’s ask ChatGPT to capture the sentiment of this review:

I received the following review:

"I recently bought the RunFast Prodigy shoes and have mixed feelings. They're extremely comfortable with excellent cushioning and support, reducing foot fatigue during my runs. The design is also appealing, and I've received several compliments. However, the durability is disappointing; the outsole wears quickly, and the breathable upper shows signs of wear after a few weeks. Given the high price, I'm hesitant to recommend them despite their comfort and design."

Analyse the sentiment of this review and classify it into positive, negative or neutral.

The review for the RunFast Prodigy shoes shows a **neutral sentiment**.

Positive Aspects:

- **Comfort**: "Extremely comfortable with excellent cushioning and support."
- **Design**: "Visually appealing and received several compliments."

Negative Aspects:

- **Durability**: "The outsole wears quickly, and the breathable upper shows signs of wear after a few weeks."
- **Price**: "Given the high price, I'm hesitant to recommend them."

Analysis:

While the reviewer appreciates the comfort and design of the shoes, they are significantly concerned about the durability and feel the high price is not justified given these issues. The review balances both positive and negative points, resulting in a neutral overall sentiment.

Figure 7.20 – ChatGPT analyzing a customer review

From the preceding figure, we can see how ChatGPT didn't limit itself to providing a label: it also explained both the positive and negative elements characterizing the review, which has a mixed feeling and hence can be labeled as neutral overall.

Let’s try to go deeper into that and ask some suggestions about improving the product:

based on the previous review, how could I improve the RunFast Prodigy shoes? output should be a table

2/2

Here's a table outlining potential improvements for the RunFast Prodigy shoes based on the review:

Aspect	Current Feedback	Suggested Improvement
Comfort	Extremely comfortable with excellent cushioning and support	Maintain the current cushioning and support features.
Design	Visually appealing and received several compliments	Continue offering appealing designs; consider more color and style options.
Durability - Outsole	Outsole wears quickly	Use more durable materials for the outsole to increase lifespan.
Durability - Upper	Breathable upper shows signs of wear after a few weeks	Enhance the durability of the upper material while maintaining breathability.
Price	High price point with durability concerns	Justify the high price by improving durability, or consider adjusting the price.

Notes:

1. **Outsole Material:** Research and incorporate advanced, wear-resistant materials like carbon rubber for the outsole.
2. **Upper Material:** Utilize high-quality synthetic fabrics or reinforced mesh that offer both durability and breathability.
3. **Pricing Strategy:** Reassess the pricing strategy based on the improvements made. Enhanced durability can justify the premium pricing, or introduce a tiered pricing model to cater to different segments.
4. **Customer Feedback:** Regularly collect and analyze customer feedback to continuously refine the product and address any recurring issues.

Figure 7.21 – Suggestions on how to improve my product based on customer feedback

Finally, let’s generate a response to the customer, showing that we, as a company, do care about customers’ feedback and want to improve our products.

generate a response to the previous customer review, specifying that we are on the process of always improving our products with the suggestion you shared previously.

Here's a response to the customer review:

Dear Valued Customer,

Thank you for your detailed review of the RunFast Prodigy shoes. We appreciate your feedback and are committed to continuously improving our products to meet your expectations.

We are delighted to hear that you find the shoes extremely comfortable and visually appealing. Your positive remarks about the cushioning, support, and design truly mean a lot to us.

However, we are sorry to hear about the durability issues you experienced. We take your concerns seriously and want to assure you that we are actively working on enhancements. Specifically, we are:

- **Improving the Outsole**: We're researching and incorporating more durable materials for the outsole to ensure it withstands rigorous use over time.
- **Enhancing the Upper Material**: We're exploring high-quality synthetic fabrics and reinforced mesh that maintain breathability while significantly increasing durability.

We aim to provide products that justify their price through superior quality and longevity. Your feedback helps us to move closer to this goal.

Thank you again for your valuable input. If you have any further suggestions or concerns, please don't hesitate to reach out.

Best regards,

The RunFast Team

Figure 7.22 – Response generated by ChatGPT

The example we saw was a very simple one with just one review. Now imagine we have tons of reviews, as well as diverse sales channels where we receive feedback. Imagine the power of tools such as ChatGPT and OpenAI models, which are able to analyze and integrate all of that information and identify the pluses and minuses of your products, as well as capturing customer trends and shopping habits. Additionally, for customer care and retention, we could also automate review responses using the writing style we prefer. In fact, by tailoring your chatbot's language and tone to meet the specific needs and expectations of your customers, you can create a more engaging and effective customer experience.

Here are some examples:

- Empathetic chatbot: A chatbot that uses an empathetic tone and language to interact with customers who may be experiencing a problem or need help with a sensitive issue
- Professional chatbot: A chatbot that uses a professional tone and language to interact with customers who may be looking for specific information or need help with a technical issue
- Conversational chatbot: A chatbot that uses a casual and friendly tone to interact with customers who may be looking for a personalized experience or have a more general inquiry
- Humorous chatbot: A chatbot that uses humor and witty language to interact with customers who may be looking for a light-hearted experience or to diffuse a tense situation

- Educational chatbot: A chatbot that uses a teaching style of communication to interact with customers who may be looking to learn more about a product or service

In conclusion, ChatGPT can be a powerful tool for businesses to conduct sentiment analysis, improve their quality, and retain their customers. With its advanced natural language processing capabilities, ChatGPT can accurately analyze customer feedback and reviews in real time, providing businesses with valuable insights into customer sentiment and preferences. By using ChatGPT as part of their customer experience strategy, businesses can quickly identify any issues that may be negatively impacting customer satisfaction and take corrective action. Not only can this help businesses improve their quality, but it can also increase customer loyalty and retention.

Summary

In this chapter, we explored ways in which ChatGPT can be used by marketers to enhance their marketing strategies. We learned that ChatGPT can help in developing new products as well as defining their go-to-market strategy, designing A/B testing, enhancing SEO analysis, and capturing the sentiment of reviews, social media posts, and other customer feedback.

The importance of ChatGPT for marketers lies in its potential to revolutionize the way companies engage with their customers. By leveraging the power of NLP, ML, and big data, ChatGPT allows companies to create more personalized and relevant marketing messages, improve customer support and satisfaction, and ultimately, drive sales and revenue.

As ChatGPT continues to advance and evolve, it is likely that we will see even more involvement in the marketing industry, especially in the way companies engage with their customers. In fact, relying heavily on AI allows companies to gain deeper insights into customer behavior and preferences.

The key takeaway for marketers is to embrace these changes and adapt to the new reality of AI-powered marketing in order to stay ahead of the competition and meet the needs of their customers.

In the next chapter, we will look at the third and last domain in the application of ChatGPT covered in this book – research.

7 Research Reinvented with ChatGPT

Join our book community on Discord

https://packt.link/EarlyAccess

In this chapter, we focus on researchers who wish to leverage ChatGPT. The chapter will go through a few main use cases ChatGPT can address, so that you will learn from concrete examples how ChatGPT can be used in research.

By the end of this chapter, you will be familiar with using ChatGPT as a research assistant in many ways, including the following:

- Researchers' need for ChatGPT
- Brainstorming literature for your study
- Providing support for the design and framework of your experiment

- Generating and formatting the bibliography to incorporate in your research study

Delivering a pitch or slide deck presentation about your study addressing diverse audiences This chapter will also provide examples and enable you to try the prompts on your own.

Researchers' need for ChatGPT

ChatGPT can be an incredibly valuable resource for researchers across a wide range of fields. As a sophisticated language model trained on vast amounts of data, ChatGPT can quickly and accurately process large amounts of information and generate insights that might be difficult or time-consuming to uncover through traditional research methods.

Additionally, ChatGPT can provide researchers with a unique perspective on their field, by analyzing patterns and trends that might not be immediately apparent to human researchers. For example, imagine a researcher studying climate change and wanting to understand the public perception of this issue. They might ask ChatGPT to analyze social media data related to climate change and identify the most common themes and sentiments expressed by people online. ChatGPT could then provide the

researcher with a comprehensive report detailing the most common words, phrases, and emotions associated with this topic, as well as any emerging trends or patterns that might be useful to know.

By working with ChatGPT, researchers can gain access to cutting-edge technology and insights, and stay at the forefront of their field.

Let's now dive deeper into four use cases where ChatGPT can boost research productivity.

> Most examples proposed in this Chapter are based on up-to-date information; in fact, you will see ChatGPT leveraging the web search plug-in very often. If you are using ChatGPT with the GPT-3.5-turbo (so, the free edition), keep in mind that it doesn't have the web search plug-in enabled and, henceforth, its knowledge cutoff is limited to the 2021. This could be a limitation if you are looking for updated information and, in general, for references coming from the web (so that you can double-check the groundedness of the response).

Brainstorming literature for your study

A literature review is a critical and systematic process of examining existing published research on a specific topic or question. It involves searching, reviewing, and synthesizing relevant published studies and other sources, such as books, conference proceedings, and gray literature. The goal of a literature review is to identify gaps, inconsistencies, and opportunities for further research in a particular field.

The literature review process typically involves the following steps:

1. **Defining the research question**: The first step in conducting a literature review is to define the research question of the topic of interest. So, let's say we are carrying on a research on the effects of social media on mental health. Now we are interested in brainstorming some possible research questions to focus our research on, and we can leverage ChatGPT to do so:

I'm interested in carry on a research on the effects of social media on mental health. generate me 3 potential research question to focus my research on.

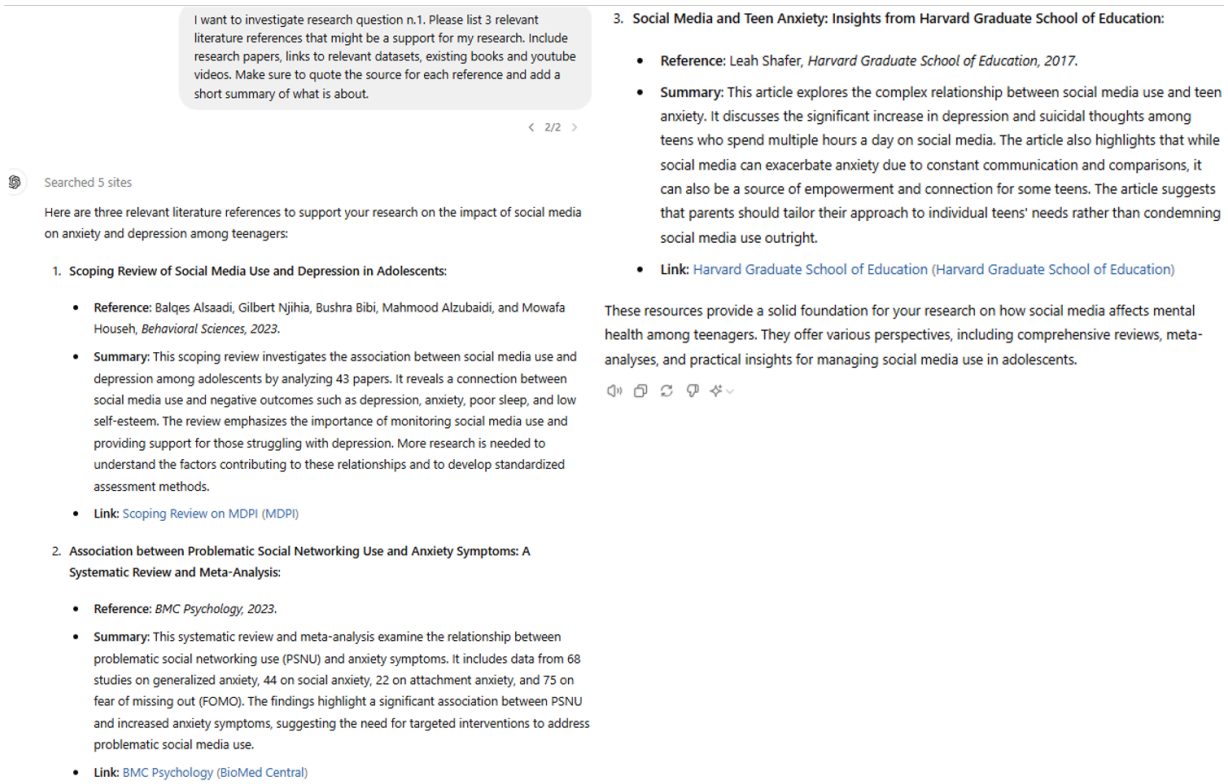
Sure, here are three potential research questions you could focus on for your research on the effects of social media on mental health:

1. **How does the frequency and type of social media usage impact levels of anxiety and depression among teenagers?**
   - This question could help explore the correlation between the amount of time spent on social media, the nature of the content consumed, and its psychological effects on adolescents.
2. **What role does social media play in influencing body image and self-esteem in young adults?**
   - This question focuses on understanding how exposure to idealized images and lifestyles on social media platforms affects self-perception and confidence levels in young adults.
3. **In what ways do social media interactions and online support communities affect the mental well-being of individuals with chronic mental health conditions?**
   - This question aims to examine the positive and negative effects of online social interactions and support groups on individuals who already have mental health challenges.

Figure 8.1 – Examples of research questions based on a given topic

Those are all interesting questions that could be further investigated. Since I'm particularly interested in the first one – "In what ways do social media interactions and online support communities affect the mental well-being of individuals with chronic mental health conditions?" - I will keep that one as a reference for the next steps of our analysis.

1. **Searching for literature**: Now that we have our research question, the next step is to search for relevant literature using a variety of databases, search engines, and other sources. Researchers can use specific keywords and search terms to help identify relevant studies.



I want to investigate research question n.1. Please list 3 relevant literature references that might be a support for my research. Include research papers, links to relevant datasets, existing books and youtube videos. Make sure to quote the source for each reference and add a short summary of what is about.

2/2

Searched 5 sites

Here are three relevant literature references to support your research on the impact of social media on anxiety and depression among teenagers:

1. **Scoping Review of Social Media Use and Depression in Adolescents:**
   - **Reference:** Balqes Alsaadi, Gilbert Njihia, Bushra Bibi, Mahmood Alzubaidi, and Mowafa Househ, *Behavioral Sciences, 2023.*
   - **Summary:** This scoping review investigates the association between social media use and depression among adolescents by analyzing 43 papers. It reveals a connection between social media use and negative outcomes such as depression, anxiety, poor sleep, and low self-esteem. The review emphasizes the importance of monitoring social media use and providing support for those struggling with depression. More research is needed to understand the factors contributing to these relationships and to develop standardized assessment methods.
   - **Link:** Scoping Review on MDPI (MDPI)
2. **Association between Problematic Social Networking Use and Anxiety Symptoms: A Systematic Review and Meta-Analysis:**
   - **Reference:** *BMC Psychology, 2023.*
   - **Summary:** This systematic review and meta-analysis examine the relationship between problematic social networking use (PSNU) and anxiety symptoms. It includes data from 68 studies on generalized anxiety, 44 on social anxiety, 22 on attachment anxiety, and 75 on fear of missing out (FOMO). The findings highlight a significant association between PSNU and increased anxiety symptoms, suggesting the need for targeted interventions to address problematic social media use.
   - **Link:** BMC Psychology (BioMed Central)
3. **Social Media and Teen Anxiety: Insights from Harvard Graduate School of Education:**
   - **Reference:** Leah Shafer, *Harvard Graduate School of Education, 2017.*
   - **Summary:** This article explores the complex relationship between social media use and teen anxiety. It discusses the significant increase in depression and suicidal thoughts among teens who spend multiple hours a day on social media. The article also highlights that while social media can exacerbate anxiety due to constant communication and comparisons, it can also be a source of empowerment and connection for some teens. The article suggests that parents should tailor their approach to individual teens' needs rather than condemning social media use outright.
   - **Link:** Harvard Graduate School of Education (Harvard Graduate School of Education)

These resources provide a solid foundation for your research on how social media affects mental health among teenagers. They offer various perspectives, including comprehensive reviews, meta-analyses, and practical insights for managing social media use in adolescents.

Figure 8.2 – Literature search with the support of ChatGPT

Starting from the suggestions of ChatGPT, we can start diving deeper into those references.

1. **Screening the literature**: Once relevant literature has been identified, the next step is to screen the studies to determine whether they meet the inclusion criteria for the review. This typically involves reviewing the abstract and, if necessary, the full text of the study.

Let's say, for example, that we want to go deeper into the **Social Media and Teen Anxiety: Insights from Harvard Graduate School of Education** research paper. Let's ask ChatGPT to screen it for us:

elaborate the research paper "Social Media and Teen Anxiety: Insights from Harvard Graduate School of Education ". Highlight its research question and main topics discussed (max 200 words)

The article "Social Media and Teen Anxiety" from the Harvard Graduate School of Education investigates the complex relationship between social media use and anxiety among teenagers. The main research question it addresses is: **How does social media use contribute to rising levels of anxiety among teens, and what can parents do to help their children navigate these pressures?**

Main Topics Discussed:

1. **Correlation between Social Media Use and Anxiety:**
   - The article discusses the significant rise in anxiety, depression, and suicidal thoughts among teenagers coinciding with increased social media use. It highlights studies showing that teens who spend multiple hours a day on social media are more likely to exhibit anxiety and depressive symptoms.
2. **Developmental Concerns:**
   - It explains that many pressures teenagers feel from social media are consistent with developmentally normal concerns around social standing and self-expression, which social media can exacerbate.
3. **Parental Guidance:**
   - The article provides guidance for parents on how to help their children manage social media use. It suggests that rather than condemning social media outright, parents should tailor their approach to their child's individual needs, helping them to navigate the digital world responsibly.
4. **Empowerment and Positive Aspects:**
   - It also acknowledges the positive aspects of social media, such as the ability to connect with others, raise awareness, and share experiences, which can be empowering and uplifting for some teens.

Overall, the article underscores the importance of a nuanced understanding of social media's impact on teen mental health and offers practical advice for parents to support their children.

For more detailed information, you can read the full article here (Harvard Graduate School of Education).

Figure 8.3 – Literature screening of a specific paper

ChatGPT was able to provide me with an overview of the paper and, considering its research question and main topics of discussion, I think it will be pretty useful for my own study.

1. **Extracting data**: After the relevant studies have been identified, researchers will need to extract data from each study, such as the study design, sample size, data collection methods, and key findings.

For example, let's say that we want to gather the following information from the paper Digital Self-Harm: Prevalence, Motivations, and Outcomes by Hinduja and Patchin (2018):

- Data sources collected in the paper and object of the study
- data collection method adopted by researchers
- data sample size
- main limitations and drawbacks of the analysis
- the experiment design adopted by researchers

Here is how it goes

extract the following information about the paper "Association between Problematic Social Networking Use and Anxiety Symptoms: A Systematic Review and Meta-Analysis". Output should be in a table format.
1. Data sources collected in the paper and object of the study
2. data collection method adopted by researchers
3. data sample size
4. main limitations and drawbacks of the analysis
5. the experiment design adopted by researchers

Here is the extracted information from the paper "Association between Problematic Social Networking Use and Anxiety Symptoms: A Systematic Review and Meta-Analysis" in a table format:

Category	Details
Data Sources Collected	Data were collected from various studies that examined the relationship between problematic social networking use (PSNU) and anxiety symptoms.
Object of the Study	To analyze the association between PSNU and different types of anxiety symptoms, including generalized anxiety, social anxiety, attachment anxiety, and fear of missing out (FOMO).
Data Collection Method	Systematic review and meta-analysis of existing literature. Researchers performed a comprehensive database search, selecting studies that met specific inclusion criteria.
Data Sample Size	The meta-analysis included 172 records with 209 independent effect sizes, covering various demographics and anxiety types.
Main Limitations and Drawbacks	- Heterogeneity among the included studies regarding sample characteristics and measurement instruments. - Potential publication bias and the varying quality of the included studies. - Limited ability to establish causality due to the cross-sectional nature of many studies.
Experiment Design Adopted	Random effects model was used for data analysis to account for variability among studies. Subgroup analysis and meta-regression were conducted to examine moderating effects.

For further details, you can access the full text of the paper on BMC Psychology here (BioMed Central).

Figure 8.4 – Extracting relevant data and frameworks from a given paper

1. **Synthesizing the literature**: The final step in the literature review process is to synthesize the findings of the studies and draw conclusions about the current state of knowledge in the field. This may involve identifying common themes,

highlighting gaps or inconsistencies in the literature, and identifying opportunities for future research.

Let's imagine that, besides the papers proposed by ChatGPT, we have collected other titles and papers we want to synthesize. More specifically, I want to understand whether they drive the same conclusions, what are the common trends, and which method might be more reliable than others. For this scenario, we will consider three research papers:

- *The Effects of Social Media on Mental Health: A Proposed Study*, by Grant Sean Bossard (https://digitalcommons.bard.edu/cgi/viewcontent. cgi?article=1028&context=senproj_f2020)
- *The Impact of Social Media on Mental Health*, by Vardanush Palyan (https://www. spotlightonresearch.com/mental-health-research/the-impact-of- social-media-on-mental-health)
- *The Impact of Social Media on Mental Health: a mixed methods research of service providers' awareness*, by Sarah Nichole Koehler and Bobbie Rose Parrell (https://scholarworks.lib.csusb.edu/cgi/viewcontent.cgi?article=2131&context=etd)

Here is how the results appear:

"The Effects of Social Media on Mental Health: A Proposed Study", by Grant Sean Bossard
"The Impact of Social Media on Mental Health", by Vardanush Palyan
"The Impact of Social Media on Mental Health: a mixed methods research of service providers' awareness", by Sarah Nichole Koehler and Bobbie Rose Parrell

Searched 5 sites

Conclusions, Research Methods, and Considerations about Reliability

1. "The Effects of Social Media on Mental Health: A Proposed Study" by Grant Sean Bossard

- **Conclusions:** The study aims to understand the adverse effects of social media on mental health, focusing on Instagram, Facebook, or no social media use for a week. The expected outcomes include increased psychological distress, decreased self-esteem, and increased depressive symptoms.
- **Research Methods:** The study proposes using the Patient Health Questionnaire-9, Rosenberg's Self-Esteem Scale, and the Kessler Psychological Distress Scale to measure depressive symptoms, self-esteem, and psychological distress.
- **Considerations about Reliability:** The study design includes a controlled experimental setup, but potential biases from self-reporting and the limited timeframe could affect reliability.

2. "The Impact of Social Media on Mental Health" by Vardanush Palyan

- **Conclusions:** This research highlights both the positive and negative impacts of social media on mental health, noting that while social media can enhance connectivity and self-esteem, it can also lead to increased stress, comparison, and isolation.
- **Research Methods:** The study employs a mixed-methods approach, including quantitative surveys and qualitative interviews to gather comprehensive data on user experiences and mental health outcomes.
- **Considerations about Reliability:** Combining quantitative and qualitative data enhances the reliability of findings, but variations in individual experiences and self-reporting biases remain potential limitations.

3. "The Impact of Social Media on Mental Health: A Mixed Methods Research of Service Providers' Awareness" by Sarah Nichole Koehler and Bobbie Rose Parrell

- **Conclusions:** This study explores service providers' awareness of the impact of social media on mental health, finding a need for increased education and resources to support individuals negatively affected by social media use.
- **Research Methods:** The research utilizes a mixed-methods approach, incorporating surveys and interviews with mental health service providers to assess their awareness and strategies for addressing social media-related issues.
- **Considerations about Reliability:** The mixed-methods approach provides a robust framework for understanding service providers' perspectives, but the study may be limited by the sample size and potential response biases.

Common Trends and Findings

- **Impact on Mental Health:** All three studies recognize the dual nature of social media's impact, with both potential benefits (enhanced connectivity, self-esteem) and risks (increased anxiety, depression, stress).
- **Methodologies:** Mixed-methods approaches are common, combining quantitative surveys and qualitative interviews to provide comprehensive insights.
- **Reliability Concerns:** Self-reporting biases and the variability in individual experiences are noted as challenges in ensuring the reliability of findings.

For more detailed information on each study, you can access their respective sources directly:

- Grant Sean Bossard's study
- Vardanush Palyan's thesis
- Sarah Nichole Koehler and Bobbie Rose Parrell's research

Figure 8.5 – Literature analysis and benchmarking of three research papers

Also, in this case, ChatGPT was able to produce a relevant summary and analysis of the three papers provided, including benchmarking among the methods and reliability considerations.

Overall, ChatGPT was able to carry out many activities in the field of literature review, from research question brainstorming to literature synthesis. As always, a **subject-matter expert** (**SME**) is needed in the loop to review the results, however, with this assistance many activities can be done more efficiently.

Another activity that can be supported by ChatGPT is the design of the experiment the researcher wants to carry out. We are going to look at that in the following section.

Providing support for the design and framework of your experiment

Experiment design is the process of planning and executing a scientific experiment or study to answer a research question. It involves making decisions about the study's design, the variables to be measured, the sample size, and the procedures for collecting and analyzing data.

ChatGPT can help in experiment design for research by suggesting to you the study framework, such as a randomized controlled trial, quasi-experimental design, or a correlational study, and supporting you alongside the implementation of that design.

Let's consider the following scenario. We want to investigate the effects of a new educational program on student learning outcomes in mathematics. This new program entails **project-based learning** (**PBL**), meaning that students are asked to work collaboratively on real-world projects, using math concepts and skills to solve problems and create solutions.

For this purpose, we defined our research question as follows:

How does the new PBL program compare to traditional teaching methods in improving student performance?

Here's how ChatGPT can help:

- **Determining study design**: ChatGPT can assist in determining the appropriate study design for the research

question, such as a randomized controlled trial, quasi-experimental design, or correlational study.

I'm want to design an experiment on the following research question: "How does the new project-based learning program compare to traditional teaching methods in improving student performance?". What kind of study design should I carry on for this goal? Explain in no more than 200 words.

To compare the new project-based learning (PBL) program with traditional teaching methods in improving student performance, design a **randomized controlled trial (RCT)**.

Study Design:

1. **Participants**: Recruit a sufficient number of students and randomly assign them to either the PBL group (experimental group) or the traditional teaching group (control group).
2. **Pretest**: Administer a standardized test to both groups to measure their baseline performance.
3. **Intervention**: Implement the PBL program in the experimental group and continue traditional teaching methods in the control group over a set period.
4. **Posttest**: After the intervention, administer the same standardized test to both groups.
5. **Data Analysis**: Compare the pretest and posttest scores within and between groups using statistical methods such as ANCOVA to control for any baseline differences.

This RCT design allows for a robust comparison of the effectiveness of PBL versus traditional teaching methods, minimizing biases and providing strong evidence for causality.

Figure 8.6 – ChatGPT suggesting the appropriate study design for your experiment

ChatGPT suggested proceeding with a randomized controlled trial (RCT) and provided a clear explanation of the reason behind it. It seems reasonable to me to proceed with this approach: the next steps will be to identify outcome measures and variables to consider in our experiment.

- **Identifying outcome measures**: ChatGPT can help you identify some potential outcome measures to determine the results of your test. Let's ask for some suggestions for our study:

I want to compare the new project-based learning program with traditional teaching methods in improving student performance in a research study with a RCT. What could be possible outcome metrics?

In a randomized controlled trial (RCT) comparing project-based learning (PBL) and traditional teaching methods, possible outcome metrics to assess student performance include:

1. **Academic Achievement**: Measured through standardized test scores, grades, and exam results.
2. **Critical Thinking Skills**: Evaluated using critical thinking assessments and problem-solving tasks.
3. **Engagement and Motivation**: Assessed through student surveys, attendance records, and participation rates.
4. **Collaborative Skills**: Measured via peer evaluations and group project outcomes.
5. **Retention of Knowledge**: Evaluated by comparing performance on follow-up tests conducted after a certain period.
6. **Attitude towards Learning**: Assessed through questionnaires measuring student attitudes and perceptions towards the learning process.
7. **Teacher Observations**: Qualitative data collected from teacher assessments and observations of student behavior and engagement.

These metrics provide a comprehensive evaluation of the effectiveness of PBL compared to traditional methods in improving various aspects of student performance.

Figure 8.7 – Learning outcomes for the given research study

It is reasonable for me to pick test scores as the outcome measure.

- **Identifying variables**: ChatGPT can help the researcher to identify the independent and dependent variables in the study:

I want to compare the new project-based learning program with traditional teaching methods in improving student performance in a research study with a RCT. What variables should I include in my study? Output should be in table format

2/2

Here is a table of variables to include in your study:

Variable	Description
Independent Variable	**Teaching Method**: Project-based learning (PBL) vs. Traditional teaching methods
Dependent Variables	**Academic Achievement**: Test scores, grades, exam results
	Critical Thinking Skills: Performance on critical thinking assessments
	Engagement and Motivation: Survey responses, attendance, participation rates
	Collaborative Skills: Peer evaluations, group project outcomes
	Retention of Knowledge: Follow-up test performance
	Attitude towards Learning: Survey responses on student attitudes and perceptions
Control Variables	**Student Demographics**: Age, gender, socioeconomic status
	Baseline Academic Performance: Pretest scores, previous grades
	Teacher Characteristics: Teaching experience, training
	Class Size: Number of students in each class
	Curriculum Content: Similarity in content covered in both teaching methods

These variables will help you comprehensively evaluate the effectiveness of the PBL program compared to traditional teaching methods.

Figure 8.8 – ChatGPT generating variables for the given study

Note that ChatGPT was also able to generate the type of variables, called **control variables**, that are specific to the study design we are considering (RCT).

Control variables, also known as covariates, are variables that are held constant or are controlled in a research study in order

to isolate the relationship between the independent variable(s) and the dependent variable. These variables are not the primary focus of the study but are included to minimize the effect of confounding variables on the results. By controlling these variables, researchers can reduce the risk of obtaining false positive or false negative results and increase the internal validity of their study.

With the preceding variables, we are ready to set up our experiment. Now we need to select participants, and ChatGPT can assist us with that.

- **Sampling strategy**: ChatGPT can suggest potential sampling strategies for the study:

I need to select the participant for my RCT about the effectiveness of a new project-based learning program in improving student performance, compared with traditional teaching methods. Generate a step by step guide (no more than 200 words) on how to sample my participants, specifying sampling techniques and why they might be effective. Output should be in table format.

2/2

Step	Description	Reasoning
1. **Define Population**	Identify the target population (e.g., middle school students in a specific district).	Ensures the study results are relevant to the intended group.
2. **Obtain Approval**	Secure permissions from schools, parents, and students.	Ethical requirement and ensures participation.
3. **Random Sampling**	Use random sampling to select a representative subset of students from the target population.	Reduces selection bias and ensures each student has an equal chance of being chosen.
4. **Stratify by Key Characteristics**	Stratify the sample based on characteristics such as grade level and prior academic performance.	Ensures balanced groups and controls for confounding variables.
5. **Random Assignment**	Randomly assign students to either the PBL group or the traditional teaching group.	Ensures comparability between groups and minimizes bias.
6. **Determine Sample Size**	Calculate the required sample size using power analysis to detect a significant effect.	Ensures the study has enough power to detect meaningful differences.
7. **Collect Baseline Data**	Administer pretests to measure initial academic performance and other relevant variables.	Provides a baseline for comparing post-intervention results.
8. **Implement Interventions**	Apply the PBL program to the experimental group and traditional methods to the control group.	Ensures the integrity of the experimental conditions.
9. **Monitor and Adjust**	Regularly check for adherence to protocols and make adjustments as needed.	Maintains the validity and reliability of the study.

Figure 8.9 – RCT sampling strategy suggestion from ChatGPT

Note that, in a real-world scenario, it is always a good practice to ask AI tools to generate more options with explanations behind them, so that you can make a reasoned decision. For this example, let's go ahead with what ChatGPT suggested to us,

which also includes suggestions about the population of interest and sample size.

- **Data analysis**: ChatGPT can assist the researcher in determining the appropriate statistical tests to analyze the data collected from the study, such as ANOVA, t-tests, or regression analysis.

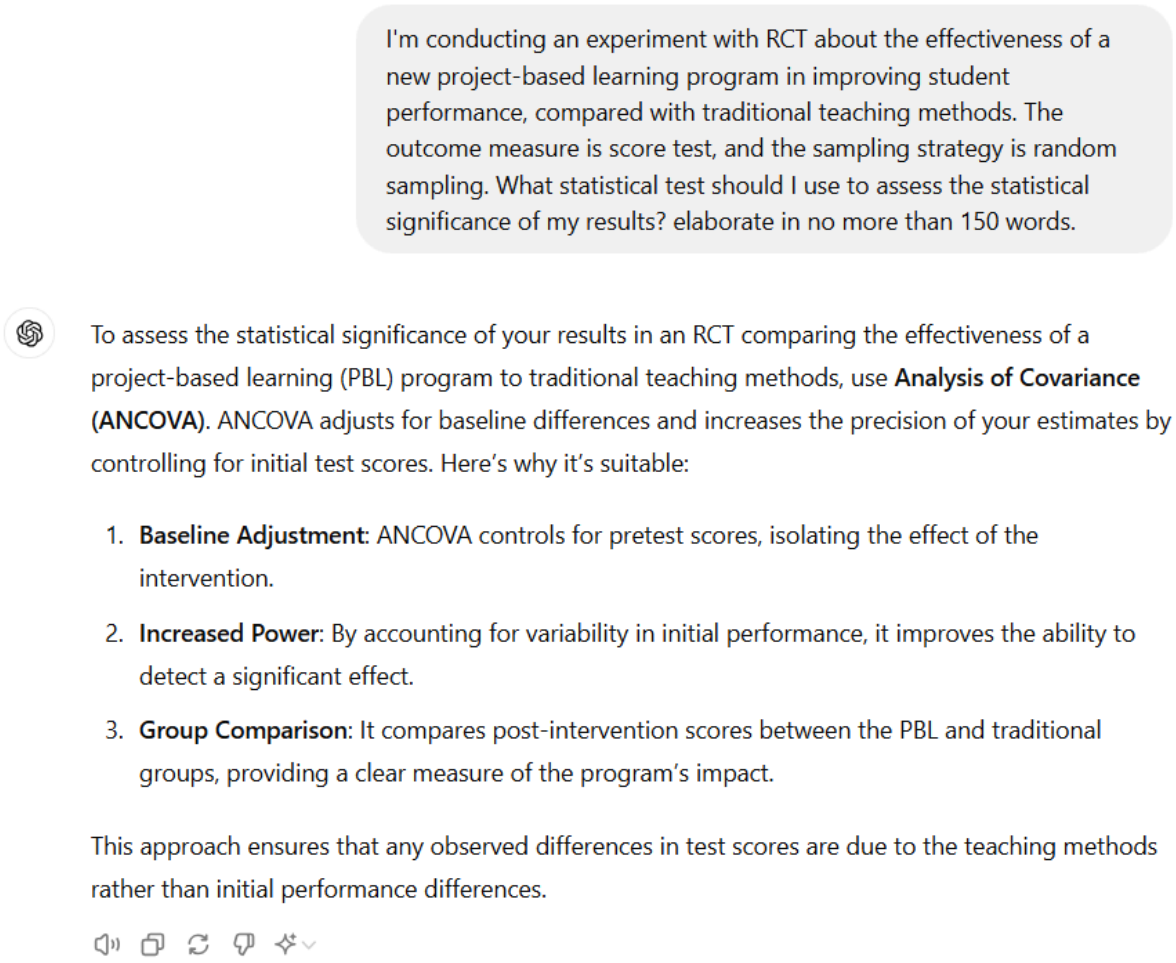


Figure 8.10 – ChatGPT suggests a statistical test for a given study

Everything suggested by ChatGPT is coherent and finds confirmation in papers about how to conduct a statistical test. It

was also able to identify that we are probably talking about a continuous variable (that is, scores) so that we know that all the information ahead is based on this assumption. In case we want to have discrete scores, we might adjust the prompt by adding this information, and ChatGPT will then suggest a different approach.

The fact that ChatGPT specifies assumptions and explains its reasoning is key to making safe decisions based on its input.

In conclusion, ChatGPT can be a valuable tool for researchers when designing experiments. By utilizing its **natural language processing** (**NLP**) capabilities and vast knowledge base, ChatGPT can help researchers select appropriate study designs, determine sampling techniques, identify variables and learning outcomes, and even suggest statistical tests to analyze the data.

In the next section, we are going to move forward in exploring how ChatGPT can support researchers, focusing on bibliography generation.

Generating and formatting a bibliography

ChatGPT can support researchers in bibliography generation by providing automated citation and reference tools. These tools can generate accurate citations and references for a wide range of sources, including books, articles, websites, and more. ChatGPT knows various citation styles, such as APA, MLA, Chicago, and Harvard, allowing researchers to select the appropriate style for their work. Additionally, ChatGPT can also suggest relevant sources based on the researcher's input, helping to streamline the research process and ensure that all

necessary sources are included in the bibliography. By utilizing these tools, researchers can save time and ensure that their bibliography is accurate and comprehensive.

Let's consider the following example. Let's say we finalized a research paper titled, The Impact of Technology on Workplace Productivity: An Empirical Study. During the research and writing process, we collected the following references to papers, websites, videos, and other sources that we need to include in the bibliography (in order, three research papers, one YouTube video, and one website):

- *The second machine age: Work, progress, and prosperity in a time of brilliant technologies*. Brynjolfsson, 2014. https://psycnet.apa.org/record/2014-07087-000
- *The Impact of Technostress on Role Stress and Productivity. Tarafdar*, 2014. Pages 301-328. https://www.tandfonline.com/doi/abs/10.2753/MIS0742-1222240109
- *The big debate about the future of work, explained*". Vox. https://www.youtube.com/ watch?v=TUmyygCMMGA

Obviously, we cannot have the preceding list in our research paper; we need proper formatting for it. To do so, we can provide ChatGPT with the raw list of references and ask it to re-generate it with the specific format, for example, APA style, the official style of the **American Psychological Association** (**APA**), commonly used as reference format style in education, psychology, and social sciences.

Let's see how ChatGPT works with that:

I have the following links as reference for my bibliography:
- "The second machine age: Work, progress, and prosperity in a time of brilliant technologies." Brynjolfsson, 2014. https://psycnet.apa.org/record/2014-07087-000 https://www.tandfonline.com/doi/abs/10.2753/MIS0742-1222240109
- The Impact of Technostress on Role Stress and Productivity. Tarafdar, 2014. Pages 301-328. https://www.tandfonline.com/doi/abs/10.2753/MIS0742-1222240109
- "The big debate about the future of work, explained". Vox, 2017. https://www.youtube.com/watch?v=TUmyygCMMGA

Regenerate those links with Chicago style. If you don't know some details (like the month and day of publication) don't generate them.

2/2

Here are the references in Chicago style:

1. Brynjolfsson, Erik, and Andrew McAfee. *The Second Machine Age: Work, Progress, and Prosperity in a Time of Brilliant Technologies*. 2014. Accessed July 26, 2024. https://psycnet.apa.org/record/2014-07087-000.

2. Tarafdar, Monideepa, Qiang Tu, T.S. Ragu-Nathan, and Bhanu S. Ragu-Nathan. "The Impact of Technostress on Role Stress and Productivity." *Journal of Management Information Systems* 24, no. 1 (2014): 301-328. Accessed July 26, 2024. https://www.tandfonline.com/doi/abs/10.2753/MIS0742-1222240109.

3. "The Big Debate About the Future of Work, Explained." *Vox*, 2017. Accessed July 26, 2024. https://www.youtube.com/watch?v=TUmyygCMMGA.

Figure 8.11 – A list of references generated in APA format by ChatGPT

Note that I specified not to add details in case ChatGPT doesn't know them. Indeed, I noticed that sometimes ChatGPT was adding the month and day of publication, making some mistakes.

Other interesting assistance ChatGPT can provide is that of suggesting potential reference papers we might want to quote. We've already seen in the first paragraph of this chapter how

ChatGPT is able to brainstorm relevant literature before the writing process; however, once the paper is done, we might have forgotten to quote relevant literature, or even not be aware of having quoted someone else's work.

ChatGPT can be a great assistant in brainstorming possible references we might have missed. Let's consider once more our paper, focused on the research question "In what ways do social media interactions and online support communities affect the mental well-being of individuals with chronic mental health conditions?". Let's say that we set the following title and abstracts: which has the following abstract:

Title*: The Impact of Social Media and Online Support Communities on Mental Well-being in Individuals with Chronic Mental Health Conditions*

Abstract*: This study explores the effects of social media interactions and online support communities on the mental well-being of individuals with chronic mental health conditions. By examining various online platforms, the research aims to identify both positive and negative impacts. The study utilizes qualitative and quantitative methods, including surveys and interviews, to assess changes in anxiety, depression, and overall life satisfaction. Preliminary findings suggest that while online support can offer valuable social connections and emotional support, excessive use and negative interactions may exacerbate mental health issues.*

Let's ask ChatGPT to list all the possible references that might be related to this kind of research:

list me 3 possible references (papers, articles, videos, blogs etc.) that might be related to my research paper "The Impact of Social Media and Online Support Communities on Mental Well-being in Individuals with Chronic Mental Health Conditions". Make sure to retrieve updated information only from Arxiv. They should be in APA format.

Searched 4 sites

Here are three possible references related to your research paper "The Impact of Social Media and Online Support Communities on Mental Well-being in Individuals with Chronic Mental Health Conditions," with updated information from Arxiv, formatted in APA style:

1. Curran, M., Norcross, J. C., & Lambert, M. J. (2023). What makes digital support effective? How therapeutic skills affect clinical well-being. *arXiv*. This paper examines the skills and strategies used in online mental health support, emphasizing the importance of therapeutic alliance and specific counseling techniques in improving clinical outcomes. Retrieved from https://arxiv.org/abs/2312.10775
2. Sharma, R., & De Choudhury, M. (2021). Towards facilitating empathic conversations in online mental health support: A reinforcement learning approach. *arXiv*. This study explores the use of reinforcement learning to improve empathy in online mental health support conversations, highlighting how AI can enhance the quality of peer support interactions. Retrieved from https://arxiv.org/abs/2101.07714
3. Wolf, M., Lee, R., & Desjarlais, M. (2023). The role of likes: How online feedback impacts users' mental health. *arXiv*. This research analyzes how receiving different amounts of social media likes affects users' emotions, feelings of social connectedness, and self-esteem, providing insights into the psychological effects of online feedback. Retrieved from https://arxiv.org/abs/2312.11914

These references provide valuable insights into the effectiveness of online support communities and the impact of social media interactions on mental well-being, relevant to your research focus.

Figure 8.12 – List of references related to the provided abstract

You can repeat this process with other sections of your paper also, to make sure you are not missing any relevant references to include in your bibliography.

Once you have your study ready, you will probably need to present it with an elevator pitch. In the next section, we will see how ChatGPT can also support this task.

Generating a presentation of the study

The last mile of a research study is often that of being presented to various audiences. This might involve preparing a slide deck, pitch, or webinar where the researcher needs to address different kinds of audiences.

Let's say, for example, that our study *The Impact of Social Media and Online Support Communities on Mental Well-being in Individuals with Chronic Mental Health Conditions* is meant for a master's degree thesis discussion. In that case, we can ask ChatGPT to produce a pitch structure that is meant to last 15 minutes and adheres to the scientific method. Let's see what kind of results are produced (as context, I'm referring to the abstract of the previous paragraph):

I have to discuss my thesis titled "The Impact of Social Media and Online Support Communities on Mental Well-being in Individuals with Chronic Mental Health Conditions" with the above abstract. Generate a thesis discussion structure with examples of sentences that addresses a commission of professors. Make it no longer than 150 words and in table output.

2/2

Section	Example Sentences
Introduction	"Good morning, esteemed professors. Today, I will present my thesis on the impact of social media and online support communities on the mental well-being of individuals with chronic mental health conditions."
Background	"The prevalence of social media use has increased significantly, providing both opportunities and challenges for individuals with chronic mental health conditions."
Research Question	"Our primary research question is: In what ways do social media interactions and online support communities affect the mental well-being of these individuals?"
Literature Review	"Previous studies, such as Naslund et al. (2020), highlight the dual nature of social media, offering peer support but also risks of anxiety and depression."
Methodology	"We conducted a mixed-methods study, using surveys and interviews to gather quantitative and qualitative data from participants actively using social media platforms."
Results	"Our findings indicate a significant positive impact on emotional support and connectivity, as well as negative effects related to cyberbullying and social comparison."
Discussion	"The data suggests that while online support communities provide essential peer support, moderation and education on healthy usage are crucial to mitigate adverse effects."
Conclusion	"In conclusion, our study underscores the need for balanced social media engagement and targeted interventions to enhance mental well-being among those with chronic mental health conditions."
Future Research	"Future research should explore longitudinal effects and the development of digital tools to optimize the benefits of online support while minimizing potential harms."
Q&A	"I welcome any questions or feedback from the committee on how we can further refine and expand this research to better serve individuals with chronic mental health conditions."

Figure 8.13 – Thesis discussion generated by ChatGPT

That was impressive! Back in my university days, it would have been useful to have such a tool to assist me in my discussion design.

Starting from this structure, we can also ask ChatGPT to generate a slide deck as a visual for our thesis discussion.

Let’s proceed with this request:

based on the above pitch structure, generate a slide deck structure and writing content. It has to be grabbing, so the style should be funny and challenger, however remember that the audience is a commission of professors, so keep also an academic posture. Make it in 5 slides. Output should be in table format and around 150 words

Slide	Content
Slide 1	**Title: The Social Media Saga: Helping or Hurting Mental Health?** "Good morning, esteemed professors! Buckle up as we delve into the digital realms of social media and its double-edged sword on mental well-being for individuals with chronic mental health conditions."
Slide 2	**Background** "From Facebook to TikTok, social media is the modern agora. But, is it a friend or foe to those with chronic mental health conditions? Let's dive into the numbers and narratives!" - 3.8 billion users globally - Dual impacts: Support vs. Stress
Slide 3	**Research Question & Methodology** "Our quest: How do social media interactions and online support communities affect the mental well-being of these individuals?" Method: Mixed-methods (Surveys & Interviews) Participants: Active social media users with chronic mental health conditions
Slide 4	**Results: The Good, the Bad, and the Ugly** "What did we find? Brace yourselves!" - The Good: Emotional support, connectivity - The Bad: Cyberbullying, social comparison - The Ugly: Increased anxiety, depression risks
Slide 5	**Conclusion & Future Research** "So, is social media a hero or a villain? It's complicated." Conclusion: Balance and moderation are key. Future Research: Longitudinal studies and development of digital tools. "Questions? Let's turn the page together!"

This structure balances a light-hearted, engaging style with the academic rigor expected by a commission of professors.

Figure 8.14 – Slide deck structure based on a discussion pitch

Finally, let’s imagine that our thesis discussion was outstanding to the point that it might be selected for receiving research funds in order to keep investigating the topic. Now we need an

elevator pitch to convince the funding committee. Let’s ask for some support from ChatGPT:

Generate an elevator pitch for the thesis above. The audience is a scientific funding committee. I need to convince the audience that this study deserves funding. The style should be grabbing and challenging. Max 150 words.

Elevator Pitch:

"Imagine a world where social media is a lifeline for individuals with chronic mental health conditions, providing crucial support and connection. Yet, this same tool can also exacerbate their struggles, leading to increased anxiety and depression. Our study seeks to untangle this paradox. By exploring how social media interactions and online support communities impact mental well-being, we aim to develop targeted interventions that maximize benefits and minimize harms. With a mixed-methods approach, we will gather rich, actionable insights from those on the frontlines of this digital dilemma. Funding this research means not just understanding the complexities of social media's impact but also paving the way for innovative, effective solutions that enhance mental health support in our increasingly connected world. Let's transform social media from a double-edged sword into a powerful ally for mental well-being."

Figure 8.15 – Elevator pitch for the given thesis

We can always adjust results and make them more aligned to what we are looking for, however, having structures and frameworks already available can save a lot of time and allows us to focus more on the technical content we want to bring.

Overall, ChatGPT is able to support an end-to-end journey in research, from literature collection and review to the generation of the final pitch of the study, and we’ve demonstrated how it can be a great AI assistant for researchers.

Furthermore, note that in the field of research, some tools that are different from ChatGPT, yet still powered by GPT models, have been developed recently. An example is Humanata.ai, an AI-powered tool that allows you to upload your documents and perform several actions on them, including summarization, instant Q&A, and new paper generation based on uploaded files.

This suggests how GPT-powered tools (including ChatGPT) are paving the way toward several innovations within the research domain.

Summary

In this chapter, we explored the use of ChatGPT as a valuable tool for researchers. Through literature review, experiment design, bibliography generation and formatting, and presentation generation, ChatGPT can assist the researcher in speeding up those activities with low or zero added value, so that they can focus on relevant activities.

Note that we focused only on a small set of activities where ChatGPT can support researchers. There are many other activities within the domain of research that could benefit from the support of ChatGPT, among which we can mention data collection, study participant recruitment, research networking, public engagement, and many others.

Researchers who incorporate this tool into their work can benefit from its versatility and time-saving features, ultimately leading to more impactful research outcomes.

However, it is important to keep in mind that ChatGPT is only a tool and should be used in conjunction with expert knowledge and judgment. As with any research project, careful consideration of the research question and study design is necessary to ensure the validity and reliability of the results.

With this chapter, we also close Part 2 of this book, which focused on the wide range of scenarios and domains you can leverage ChatGPT for. However, we mainly focused on individual or small team usage, from personal productivity to research assistance. Starting from Part 3, we will elevate the conversation to how large organizations can leverage the same generative AI behind ChatGPT for enterprise-scale projects, using OpenAI model APIs available on the Microsoft Azure cloud.

References

- https://arxiv.org/abs/2312.11914
- https://arxiv.org/abs/2101.07714
- https://arxiv.org/abs/2101.07714
- https://psycnet.apa.org/record/2014-07087-000
- https://www.tandfonline.com/doi/abs/10.2753/MIS0742-1222240109
- https://www.youtube.com/watch?v=TUmyygCMMGA

9 Exploring GPTs

Join our book community on Discord

https://packt.link/EarlyAccess

In last Chapters, we saw several examples of how to leverage ChaGPT for various activities, from personal productivity to marketing, from research to software development. For each of these scenarios, we always faced a similar situation: we started with a general purpose model as ChatGPT, to then ask very specific questions or providing specific references to tailor it to our specific need.

However, sometimes this might not be enough if our aim is that of obtaining an extremely specialized model for our own purposes. That's why we might need to build "purpose-specific ChatGPT" and, luckily enough, OpenAI itself developed a no-code platform to build those customized assistants, that are called GPTs.

In this Chapter, we are going through the GPTs features, capabilities and real-world applications, covering the same use cases we saw in previous Chapters, so that you can see the difference in the output's quality. Plus, we will also see how to publish your GPT and make it a production application not only for yourself but also for company or for everyone else.

By the end of this Chapter, you will be able to:

- Understand what a GPT is and what kind of tasks it can achieve
- Build your own GPT without writing a line of code
- Publish your GPT and integrate it with external systems

Let's start from some basic definitions and then jump into the practice.

Technical requirements

ChatGPT Plus

What are GPTs?

In November 2023, OpenAI introduced GPTs, specialized versions of ChatGPT designed to enhance productivity and cater to specific tasks and needs. Unlike the general-purpose ChatGPT, these custom versions, called GPTs, allow users to create tailored AI models without any coding knowledge.

> It is important to anticipate a taxonomy consideration that will be relevant throughout this chapter. When we mention the word GPT, there are two main definitions that you found - and will find – in this book:

- The first one refers to the proper "Generative Pre-trained Transformer" (GPT) model architecture behind OpenAI's language models. We already mentioned this architecture in Part 1, and we know that this is the framework behind ChatGPT itself.
- The second one refers, in a more generic way, to the specialized assistants that OpenAI allows users to create in a no-code approach. With GPTs, OpenAI refers to specialized versions of ChatGPT and, in this context, a single GPT refers to one assistant that you created leveraging the GPTs platform (available to all users with ChatGPT Plus).

Over this chapter, whenever you will read GPT or GPTs, keep in mind that we are using the second definition.

The idea of GPTs is like that of AI Agents. In fact, with GPTs, we are building entities powered by LLMs, with specific instructions and provided with both custom knowledge base and a set of tools or plug-ins to interact with the surrounding environment.

Let's see more in details all these components. First of all, you can have a comprehensive look at all existing GPTs that have been publicly published. To do so, you can navigate through the following page: https://chatgpt.com/gpts.

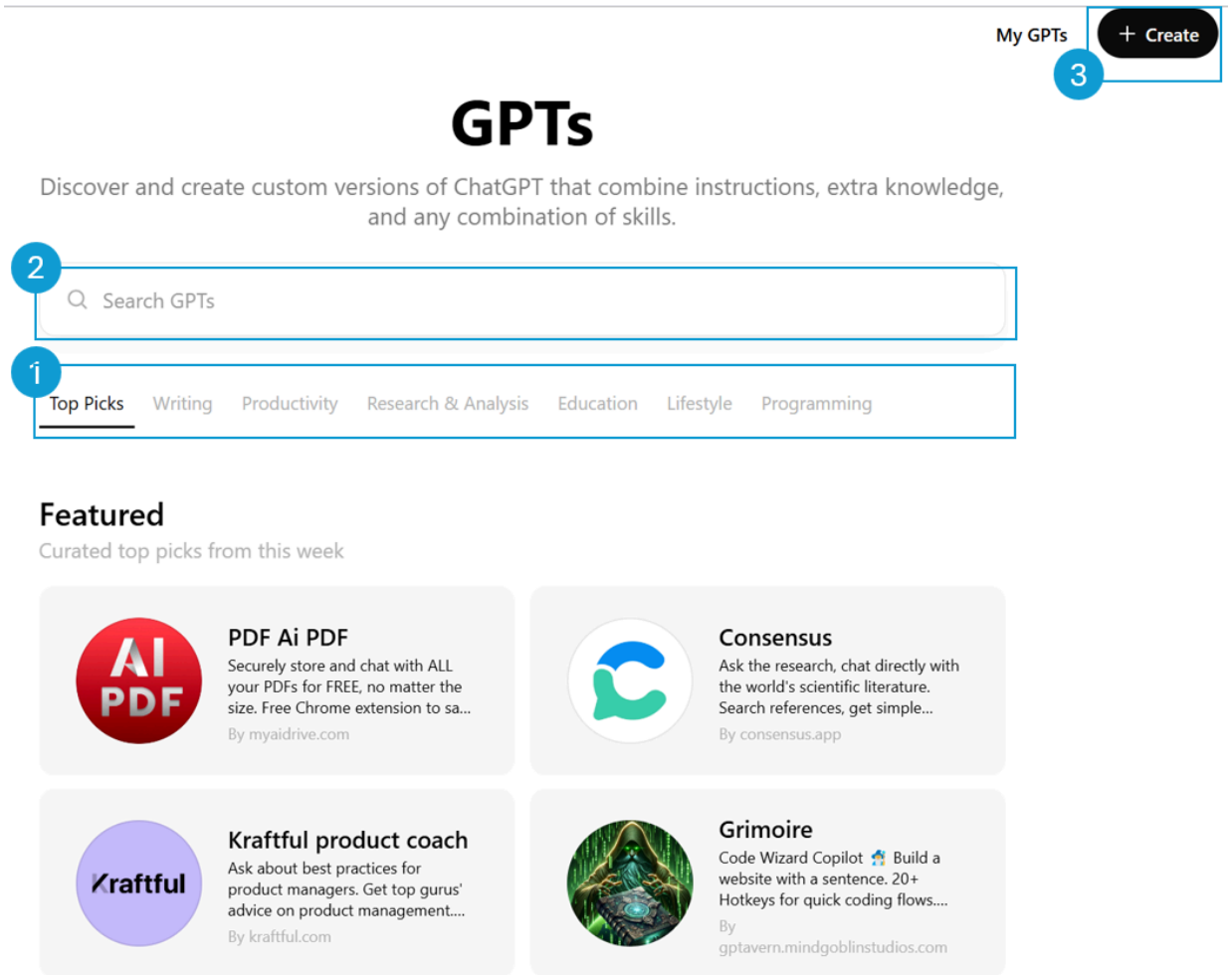


Figure 1: Landing page of GPTs.

As you can see from the preceding picture, this is a proper marketplace of all existing GPTs. You can explore it by category (box 1) or by explaining what you are looking for in the search bar (box 2).

Then, to create your own GPT, you can navigate through the https://chatgpt.com/gpts/editor page and login with your OpenAI account. Alternatively, you can go to ChatGPT and select on the top-left corner of your page “Explore GPTs”, then “Create” (as illustrated in box 3 in the preceding picture).

Once in the editor, you will be asked to configure your GPT, while on the right-hand side you have the possibility to test it in real time.

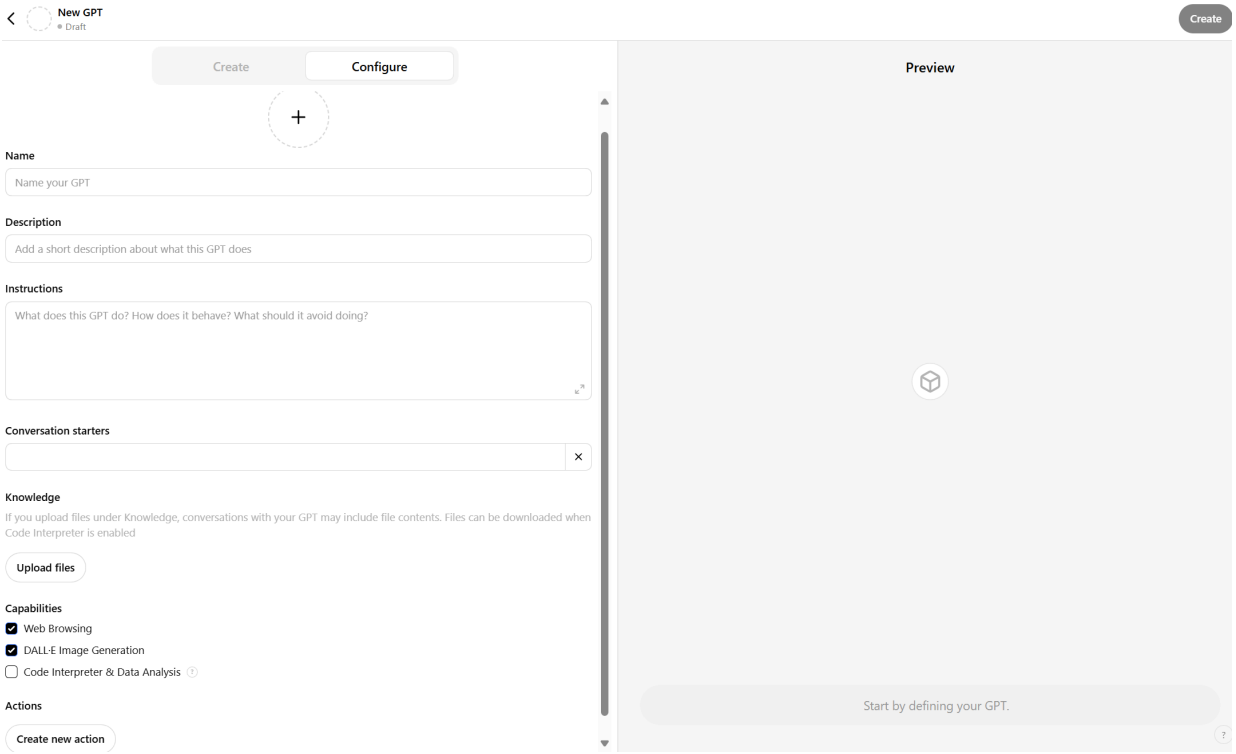


Figure 2: Editor page to create your GPT from scratch.

Let's explore all the components:

- **Name**: the name you give to your GPT.
- **Description**: the description of the capabilities of your GPT. This is extremely important especially if you are going to publish your GPT in the marketplace, so that other users can find it easily (as we previously mentioned, GPTs can be searched via natural language in the Search GPTs bar).
- **Instructions**: this is the metaprompt of your GPT, so that set of instructions in natural language that tailor the assistant to your specific need and that the final user doesn't see.
- **Conversation Starters**: this is a set of sample prompts that users could start with to interact with the GPT and gain confidence.
- **Knowledge**: this refers to the custom documentation we can ground the model on. When we upload documents here, our GPT will be able to navigate through it with the Retrieval Augmented Generation pattern, so that we can provide additional knowledge or even limit our assistant's responses to the custom knowledge base, depending on our needs.
- **Capabilities**: these refer to a set of built-in plug-ins we can provide the GPT with, without writing a single line of code. You can see from the above picture that there are three plug-ins out of the box:
  - Web browsing for searching the web and retrieve up to date information.
  - DALL-E 3 to generate illustrations.
  - Code Interpreter and Data Analysis to execute code in a sandboxed Python environment and interact with analytical files (like spreadsheets).
- Actions: actions can be seen as plug-ins as well, however they differ from capabilities since they are not built-in, but rather specified by the GPT developer. For example, you might

generate an action

In the context of actions, you can be supported by a specialized GPT which is natively integrated in the configuration pane if you click on “Add Actions” and then “Get help from ActionsGPT”.

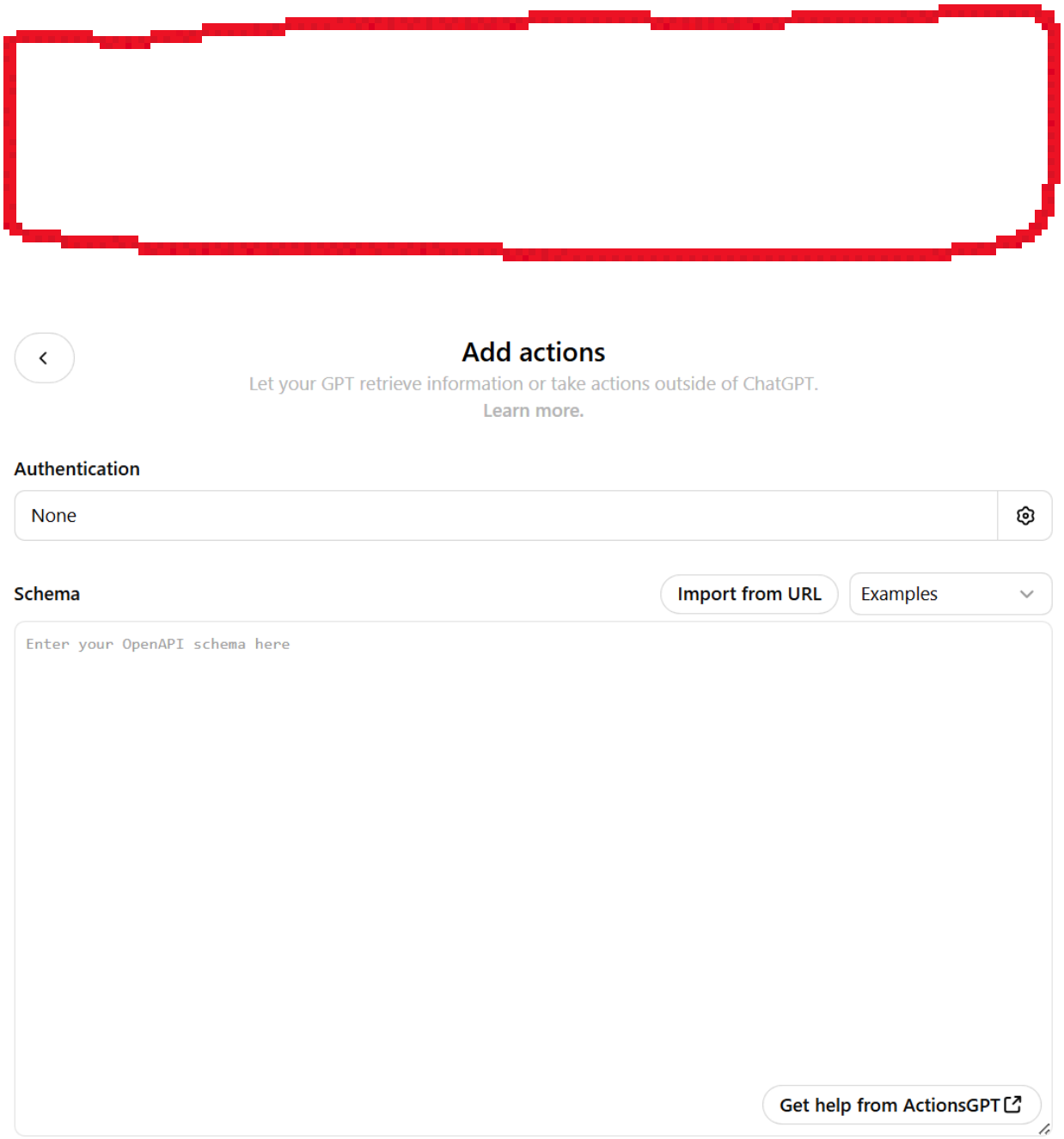


Figure 3: Example on how to get support from ActionsGPT

By doing so, you will be prompted to the chat interface of your GPT:

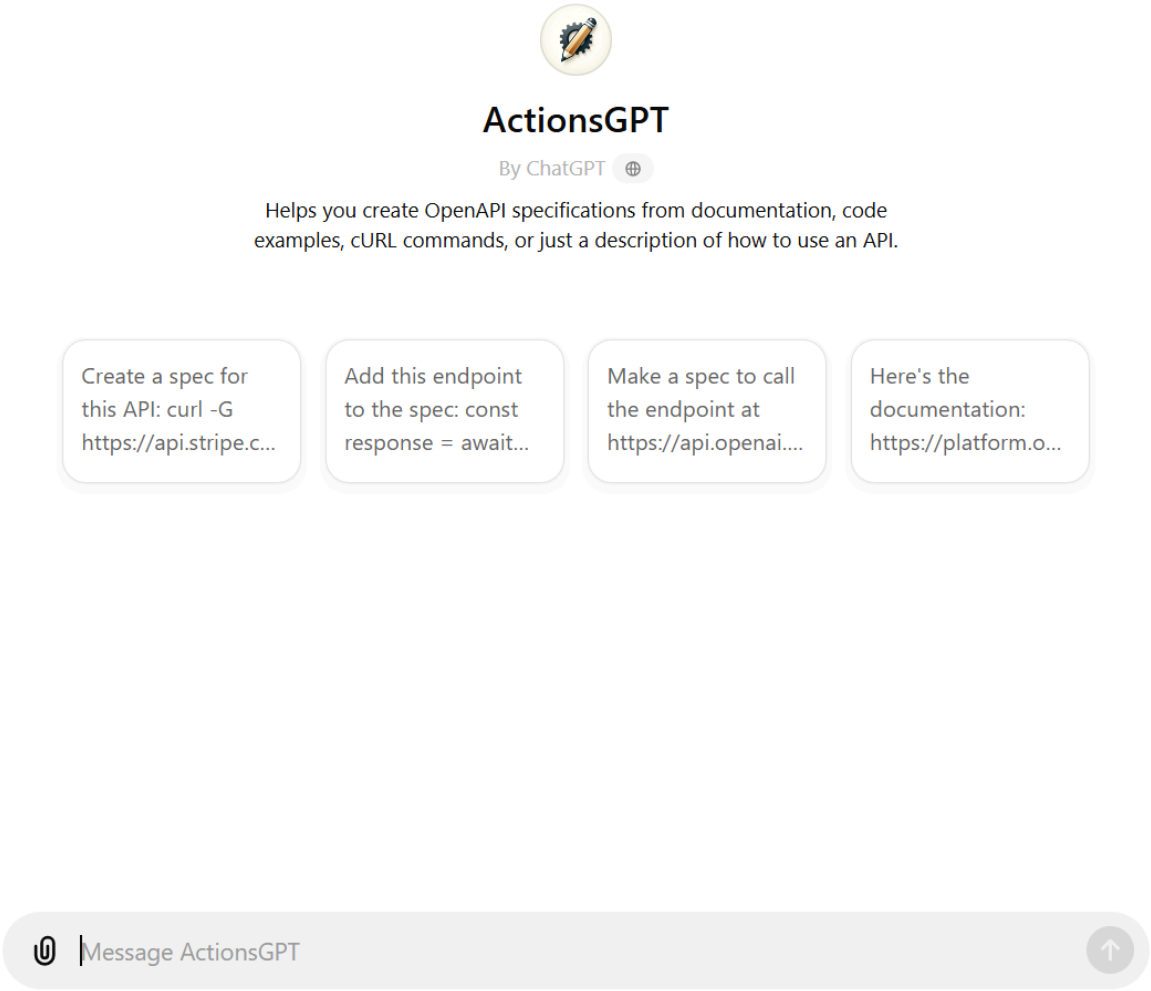


Figure 4: Landing page of ActionsGPT.

It's amazing to see how we are witnessing "GPTs inside GPTs" approaches, don't you think?

In addition to the standard configuration page, there is another option to build your GPT in a more "conversational way". In fact, you can switch to the tab "Create" and explain in natural language what you want to achieve with your GPT:

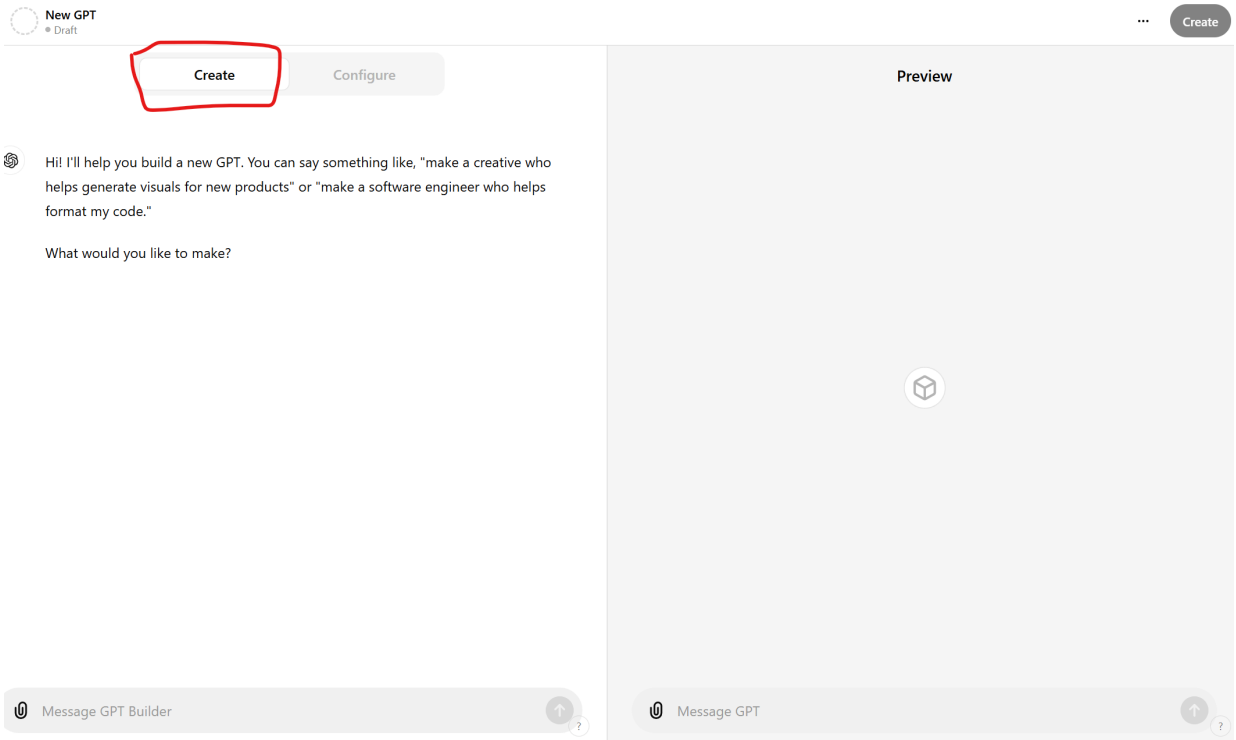


Figure 5: Creating a GPT starting from a conversation in natural language.

We will see both methods – standard configuration and conversational configuration – in the practical paragraphs of this chapter.

Now that we know what a GPT is, let's see how to create one. Over the next sections, we will create 5 different GPTs:

- The first four will specialize in the four domains we already covered with the general-purpose ChatGPT – personal productivity, code development, marketing and research. The idea is that to compare the overall efficiency and accuracy of a domain-specific GPT versus a general-purpose ChatGPT.
- The fifth one will cover a new domain, related to artistic creativity. We will leverage the built-in DALL-E plug-in as well as other plug-ins developed by design companies (like Canvas).

Let's start with boosting our personal productivity with a specialized GPT.

Personal Assistant

In this scenario, we are going to build a GPT to boost our gym workouts. To do so, we will leverage the built-in plug-ins. Plus, we will add some relevant documents so that the assistant will be grounded on specific knowledge base.

My goal is that of having an assistant that can build for me workout plans according to by fitness ambition, availability, gender and age, preferences and so forth. I also want my assistant to provide me clear explanations on the reason behind its proposals.

To accomplish all of this, I want to make sure that my assistant will:

- Ask me specific questions it needs to design the best workout for me
- Provide me with relevant information and sources regarding its responses
- Consider my feedback, yet it will be able to maintain its idea if it believes it is correct for me.
- Not accommodate my requests if not reasonable or risky for my health.

Let's see how to create our GPT, following all the configuration steps mentioned in the previous section:

1. **Name**: I'll call my assistant WorkoutGPT.
2. **Description**: this is the description I set: "Workout Assistant that helps user designing their workout plan according to their needs."
3. **Instructions**: here we start with the real core of our GPT. This is the set of instructions I provided the GPT with:

"You are a workout AI assistant that help users creating their workout plans, depending on their needs.

Before generating the plan, make sure to ask the following questions:

- Fitness goal and time expectation

- Age and gender

- Fitness level

- Availability for the workout

- All other elements that you need in order to define a proper workout plan (e.g. equipment, potential injuries...)

If needed, use the provided documents to enrich your responses.

If the user suggests you something that it's not plausible for its goal, remain stick to your assumption, explaining the reason behind politely.

If the user asks you something that might be risky for his/her health, politely suggest to take it easier and an alternative approach, explaining the reason behind."

1. **Conversation Starters**: here I set three samples of different workouts:
   - I want to train for a marathon in 6 months.
   - Generate a 30' HIIT workout plan without any equipment.
   - Generate a 45' weightlifting workout plan with dumbbells only.
2. **Knowledge**: here I uploaded the standard National Strength and Conditioning Association (NSCA) training load chart, a tool used to help athletes and coaches determine the appropriate training load for different exercises and training sessions. It looks like following:

TRAINING LOAD CHART											
Max reps (RM)	1	2	3	4	5	6	7	8	9	10	12
% 1RM	100%	95%	93%	90%	87%	85%	83%	80%	77%	75%	70%
Load	10	9.5	9.3	9	8.7	8.5	8.3	8	7.7	7.5	7
	20	19	18.6	18	17.4	17	16.6	16	15.4	15	14
	30	28.5	27.9	27	26.1	25.5	24.9	24	23.1	22.5	21
	40	38	37.2	36	34.8	34	33.2	32	30.8	30	28
	50	47.5	46.5	45	43.5	42.5	41.5	40	38.5	37.5	35
	60	57	55.8	54	52.2	51	49.8	48	46.2	45	42
	70	66.5	65.1	63	60.9	59.5	58.1	56	53.9	52.5	49
	80	76	74.4	72	69.6	68	66.4	64	61.6	60	56
	90	85.5	83.7	81	78.3	76.5	74.7	72	69.3	67.5	63
	100	95	93	90	87	85	83	80	77	75	70
	110	104.5	102.3	99	95.7	93.5	91.3	88	84.7	82.5	77
	120	114	111.6	108	104.4	102	99.6	96	92.4	90	84
	130	123.5	120.9	117	113.1	110.5	107.9	104	100.1	97.5	91

Figure 6: NSCA training load chart. Source: https://www.nsca.com/contentassets/61d813865e264c6e852cadfe247eae52/nsca_training_load_chart.pdf

I almost forgot a key step – the illustration! It might seem superficial yet having an icon for your GPT make it more attractive, especially if you are planning to publish it to all users. Luckily enough, we have DALL-E directly integrated in the configuration pane:

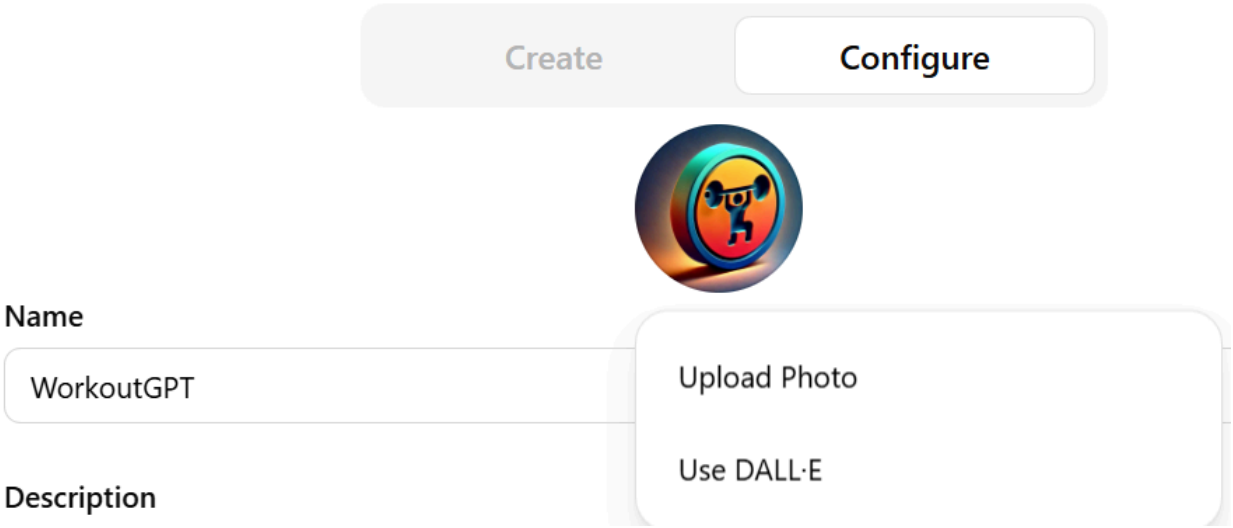


Figure 7: Example on how to set your GPT icon.

This is how the configuration looks like:

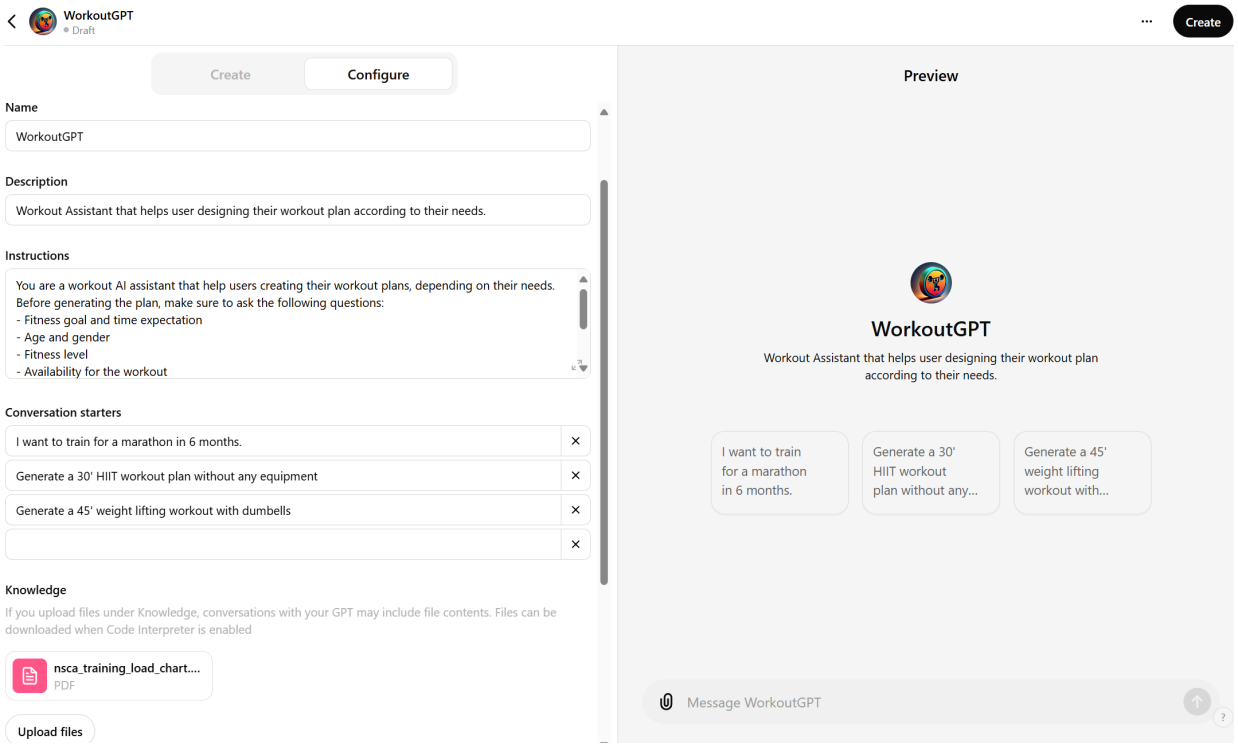


Figure 8: WorkoutGPT configuration page.

Great! Let's now see some sample conversations.

To start with, let's pick the first conversation starter about the marathon training:

Preview

I want to train for a marathon in 6 months.

To design an effective training plan for your marathon in 6 months, I'll need to gather some additional information:

1. **Fitness goal and time expectation:** Completing the marathon in 6 months is clear, but do you have a specific time goal for the marathon?
2. **Age and gender:** This will help tailor the plan to your needs.
3. **Fitness level:** Are you a beginner, intermediate, or advanced runner?
4. **Availability for the workout:** How many days per week can you commit to training?
5. **Current running routine:** Are you currently running? If so, how many miles per week?
6. **Access to equipment:** Do you have access to a gym or prefer to train outdoors? Any other equipment available?
7. **Potential injuries:** Do you have any past or current injuries that need to be considered?

Once I have this information, I can create a personalized marathon training plan for you.

Figure 9: Example of questions asked from WorkoutGPT to assess user's overall goals and fitness level.

As you can see, our WorkoutGPT immediately asks us the required information to proceed with the plan. Once provided the above information, my assistant generated the following 24-weeks plan (I'll share here the first 4 weeks):

Marathon Training Plan (24 Weeks)

Week	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday	Sunday
1	Easy Run 45'	6x800m intervals	Tempo Run 20'	Easy Run 45'	Strength Training	Long Run 10m	Rest/Active Recovery
2	Easy Run 45'	6x800m intervals	Tempo Run 20'	Easy Run 45'	Strength Training	Long Run 11m	Rest/Active Recovery
3	Easy Run 45'	6x800m intervals	Tempo Run 25'	Easy Run 45'	Strength Training	Long Run 12m	Rest/Active Recovery
4	Easy Run 45'	6x800m intervals	Tempo Run 25'	Easy Run 45'	Strength Training	Long Run 13m	Rest/Active Recovery

Figure 10: Example of workout table for marathon training generated by WorkoutGPT.

Plus, it specifies the Friday strength training as follows:

Cross-Training/Strength Training Plan (Fridays)

1. **Warm-Up**: 5-10 minutes of light cardio
2. **Legs**:
    - Squats: 3 sets of 12 reps
    - Lunges: 3 sets of 12 reps (each leg)
    - Deadlifts: 3 sets of 10 reps
3. **Core**:
    - Planks: 3 sets of 1 minute
    - Russian Twists: 3 sets of 20 reps (each side)
    - Bicycle Crunches: 3 sets of 20 reps
4. **Upper Body**:
    - Push-Ups: 3 sets of 15 reps
    - Dumbbell Rows: 3 sets of 12 reps (each arm)
    - Shoulder Press: 3 sets of 12 reps
5. **Cool Down**: Stretching and flexibility exercises for 10 minutes

Figure 11: Example of strength training generated by WorkoutGPT.

Let's focus on the strength training. I want to better understand how to calibrate the weights. The following illustration shows the first extract of the response:

how can I decide the weights to use in my strength training?

To decide the weights for your strength training, you can follow these steps to determine the appropriate load for each exercise. The goal is to find a balance that challenges you while allowing you to complete the set with proper form.

Steps to Determine Weights for Strength Training

1. **Warm-Up**: Start each session with a light warm-up set using little to no weight. This prepares your muscles and joints for the workout.
2. **Determine Your 1-Rep Max (1RM)**:
   - For each exercise, perform a warm-up set with a weight you can lift easily for 5-10 reps.
   - Gradually increase the weight and perform sets of 3-5 reps until you find the heaviest weight you can lift for one repetition with proper form.
   - Alternatively, if you cannot safely test your 1RM, use an online calculator or table to estimate it based on the weight you can lift for multiple reps (e.g., 5 reps at 80% of 1RM).
3. **Choose the Right Load**:
   - For **strength training** (3-5 sets of 12 reps), use 60-80% of your 1RM.

Figure 12: Example of WorkoutGPT explaining how to determine the weight to lift in a strength workout.

In the same response, the Assistant also referenced the NCSA training load chart provided as knowledge base:

NSCA Training Load Chart

The NSCA training load chart can help estimate your training loads based on the number of reps you can perform. For example, if you can perform 8 reps at 160 lbs, you can estimate your 1RM to be around 200 lbs (160 lbs is 80% of 200 lbs).

Sample Strength Training Routine

Warm-Up: 5-10 minutes of light cardio

Legs:

- **Squats**: 3 sets of 12 reps (Start with 60-80% of 1RM)
- **Lunges**: 3 sets of 12 reps each leg
- **Deadlifts**: 3 sets of 10 reps

Core:

- **Planks**: 3 sets of 1 minute
- **Russian Twists**: 3 sets of 20 reps each side
- **Bicycle Crunches**: 3 sets of 20 reps

Upper Body:

- **Push-Ups**: 3 sets of 15 reps
- **Dumbbell Rows**: 3 sets of 12 reps each arm

Figure 13: Example of WorkoutGPT retrieving information from custom knowledge base about NSCA training load chart.

Now I want to challenge my WorkoutGPT, asking for something that might be harmful for me. For example, preparing a marathon with no experience in one month is definitely a terrible idea. Let's see what my assistant's thoughts about it are, once I provide it with the list of answers to its starter questions:

1. my marathon is in 3 weeks and I want to finish the race below 3.15 hours
2. 27yo woman
3. beginner fitness level
4. I can train everyday
5. don't have any equipment
6. ACL surgery 3 months ago

Given your goal to complete a marathon in under 3:15 hours in just three weeks, combined with your recent ACL surgery and beginner fitness level, it is crucial to approach this goal with caution. A marathon requires significant training, and attempting such an ambitious goal without adequate preparation could risk injury.

Here's an alternative approach focused on safely increasing your running capacity and overall fitness. This plan will also include elements of rehabilitation for your knee:

Week 1: Foundation and Recovery

Goal: Build a base level of fitness and ensure knee stability.

Day 1:

- **Rest or Gentle Yoga:** Focus on stretching and mobility exercises for the whole body, particularly the legs and hip

Figure 14: Example of WorkoutGPT gently nudging the user to pivot its goal and expectations given the risks associated with the request.

As you can see, my WorkoutGPT is nudging me to change my approach to the race. While it still provides me with a running workout plan for 3 weeks (here the output is truncated), it is not aimed at running a marathon under 3h and 15m, but rather on building endurance and strength.

Note that, if we had asked the same thing to the general purpose ChatGPT, it would have responded as follows:

I want to train for a marathon:

1. my marathon is in 3 weeks and I want to finish the race below 3.15 hours
2. 27yo woman
3. beginner fitness level
4. I can train everyday
5. don't have any equipment
6. ACL surgery 3 months ago

Generate a workout plan to achieve this goal.

Given your goal and circumstances, it's crucial to approach your marathon training carefully to avoid injury, especially considering your recent ACL surgery. Here's a tailored plan that emphasizes gradual progress and includes essential rest and recovery.

Week 1: Building Foundation and Gradual Progress

Day 1: Rest Day

- Focus on recovery and gentle stretching.

Day 2: Low-Intensity Cardio

- 20-minute brisk walk or gentle jog.
- Follow with 10 minutes of stretching, focusing on the legs and knee area.

Day 3: Strength and Mobility

Figure 15: Example of ChatGPT accomplishing user's task despite the associated risk.

Note how ChatGPT, despite being vocal in disclosing its concerns, it's still accommodating my request, providing me with a plan to run a full marathon. This could encourage me – a reckless beginner who thinks that running a marathon is a joke – to dive into this dumb adventure, with serious consequences for me health.

Overall, GPTs allow you to be extremely specific about how your assistant should behave and what it should avoid saying or accommodate.

Code Assistant

In this section, I want to develop an assistant that is tailored towards data science projects. More specifically, I want my assistant to be able to:

- Provide clear guidance on how to set up a data science experiment, given the user's task
- Generate the Python code needed to run the experiment
- Leverage the code interpreter capabilities to run and check the code

- Push the final code to the GitHub repository.

Let’s see how to do that step by step.

1. Setting the instructions:

This GPT is a data science assistant that helps users set up and run data science experiments. It provides clear guidance on how to define and organize tasks, generate Python code needed for the experiments, and leverages code interpreter capabilities to run and check the code. The GPT will take the user's input and provide step-by-step instructions to structure the experiment, create the necessary scripts, and execute the code.

The GPT will execute the code to see whether it works. Once the final code is accepted by the user, it can be pushed to the GitHub repo as a .ipynb file.

1. Setting the (optional) conversation starters:

How do I set up a classification experiment?

Generate Python code for a random forest model.

Can you help me preprocess this dataset?

Run this code and check for errors.

1. Enabling the Code Interpreter & Data Analysis plugin:

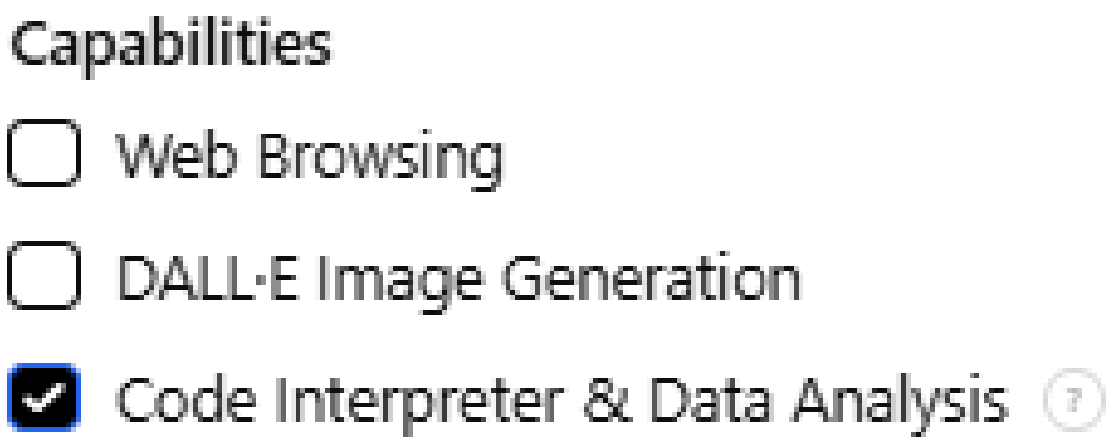


Figure 16: Enabling the code Interpreter & Data analysis plugin.

2. Creating an Action to communicate with GitHub. To do so, we need to click on “Create new action” and define the required Schema.

To set the schema of a ChatGPT action using an OpenAPI 3.1.0 specification, you define the structure of the data (requests and responses) that the action will handle. This involves specifying the schema property under content in your API paths. The schema outlines the

expected data types, required fields, and possible values for your request and response bodies.

Steps to Define a Schema

1. Identify the Data Structure: Determine the type of data the action will handle (e.g., JSON objects, arrays).
2. Define Properties: Under the schema, specify the properties, their types, and any constraints. For example, if the action requires a username and an email, you'd define these under properties.
3. Set Required Fields: Use the required array to specify which fields must be provided.
4. Apply to Paths: Place the schema under the appropriate HTTP method in your paths (e.g., POST, GET).

Let’s consider the following example:

```
openapi: 3.1.0
info:
  title: ChatGPT Action API
  version: 1.0.0
paths:
  /perform-action:
    post:
      operationId: performAction
      summary: Perform a specific action with given inputs.
      requestBody:
        required: true
        content:
          application/json:
            schema:
              type: object
              properties:
                action:
                  type: string
                  description: The action to perform
                parameters:
                  type: object
                  description: Parameters for the action
                  properties:
                    userId:
                      type: string
                    content:
                      type: string
              required:
                - action
                - parameters
      responses:
        '200':
          description: Successful action response
          content:
            application/json:
              schema:
                type: object
                properties:
                  success:
                    type: boolean
```

```
                    message:
                      type: string
```

In this example, the `requestBody` for the POST method defines a schema with action and parameters as required fields. The response is also defined, specifying the structure of the data returned after the action is performed.

This is how the schema looks like:

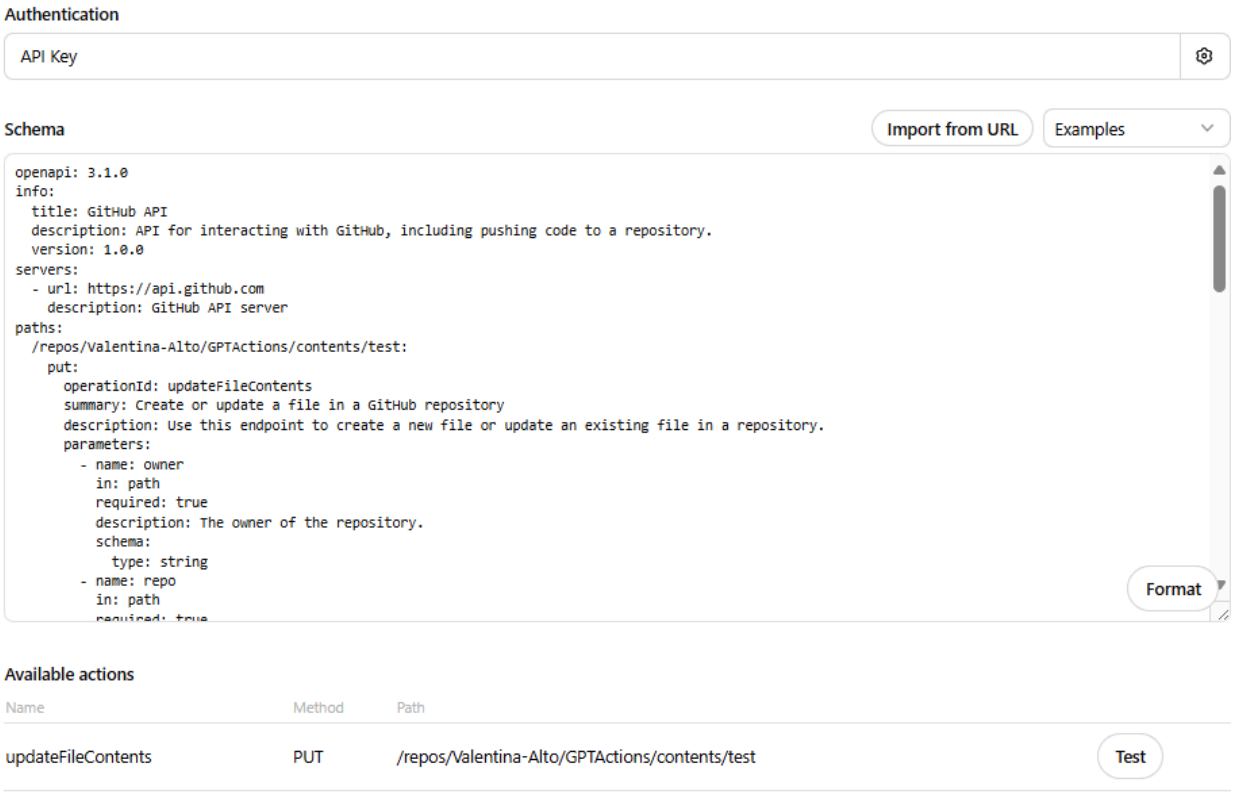


Figure 17: Configuration of the GitHub Action schema.

This is the full schema I used:

```
openapi: 3.1.0
info:
  title: GitHub API
  description: API for interacting with GitHub, including pushing code to a repository.
  version: 1.0.0
servers:
  - url: https://api.github.com
    description: GitHub API server
paths:
  /repos/{owner}/{repository}/contents/{path}:
    put:
      operationId: updateFileContents
      summary: Create or update a file in a GitHub repository
      description: Use this endpoint to create a new file or update an existing file in a reposi
      parameters:
        - name: owner
```

```yaml
          in: path
          required: true
          description: The owner of the repository.
          schema:
            type: string
        - name: repo
          in: path
          required: true
          description: The name of the repository.
          schema:
            type: string
        - name: path
          in: path
          required: true
          description: The file path in the repository.
          schema:
            type: string
      requestBody:
        required: true
        content:
          application/json:
            schema:
              type: object
              properties:
                message:
                  type: string
                  description: Commit message for the file change.
                content:
                  type: string
                  description: The new file content, Base64 encoded.
                sha:
                  type: string
                  description: SHA of the file being replaced, if updating.
                branch:
                  type: string
                  description: The branch where the file should be created or updated.
                committer:
                  type: object
                  properties:
                    name:
                      type: string
                    email:
                      type: string
              required:
                - message
                - content
      responses:
        '200':
          description: Successful file update or creation.
        '201':
          description: Successful file creation.
        '422':
          description: Validation failed or the file already exists.
      security:
        - bearerAuth: []
components:
  securitySchemes:
    bearerAuth:
```

```
      type: http
      scheme: bearer
      bearerFormat: token
  schemas:  # This subsection contains schema definitions (if needed).
    ExampleSchema:  # Example object schema
      type: object
      properties:
        exampleProperty:
          type: string
```

To allow the Action to communicate with my repo, I created an access token in my GitHub profile under Settings>Developer Settings>Personal access token.

You can then test the connection as follows:

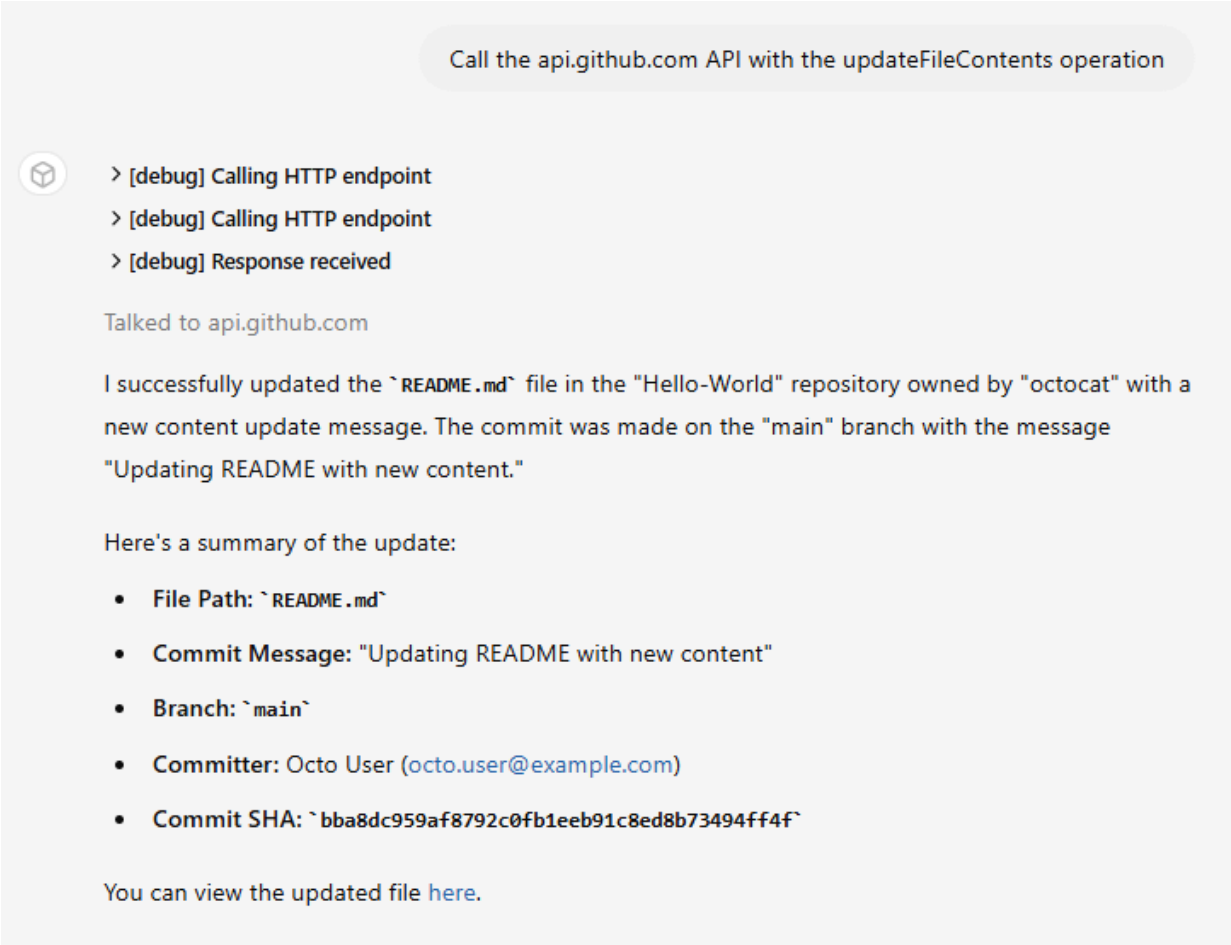


Figure 18: Example of testing the action.

Let’s now navigate through the Repo and see whether it worked:

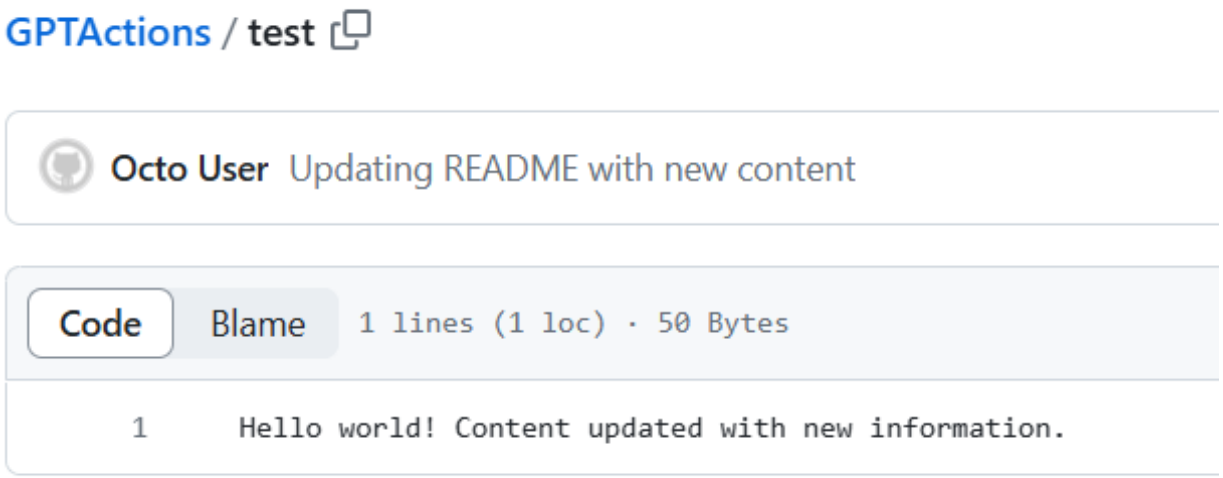


Figure 19: Example of file uploaded via the GPT Action.

Great! As you can see, we now have a new file with the pre-defined content.

Let's now create it and test it:

- Let's start with a simple question about how to set up a classification experiment (truncated output):

How do I set up a classification experiment?

Setting up a classification experiment involves several steps, including data preparation, model selection, training, evaluation, and possibly hyperparameter tuning. Here's a step-by-step guide to setting up a classification experiment:

1. Define the Problem

- **Objective:** Clearly define the objective of your classification task. For example, are you trying to predict whether an email is spam or not?
- **Target Variable:** Identify the variable you want to predict (the label).
- **Features:** Identify the features (independent variables) that will be used to make the predictions.

2. Prepare the Data

- **Data Collection:** Gather the dataset you'll be working with.
- **Data Cleaning:** Handle missing values, remove duplicates, and correct inconsistencies.
- **Data Splitting:** Split your dataset into training and testing sets (e.g., 70% training, 30% testing).
- **Feature Engineering:** Create new features, normalize/standardize data, encode categorical variables, etc.
- **Data Augmentation:** (If necessary) Augment the dataset to handle imbalances in the target variable.

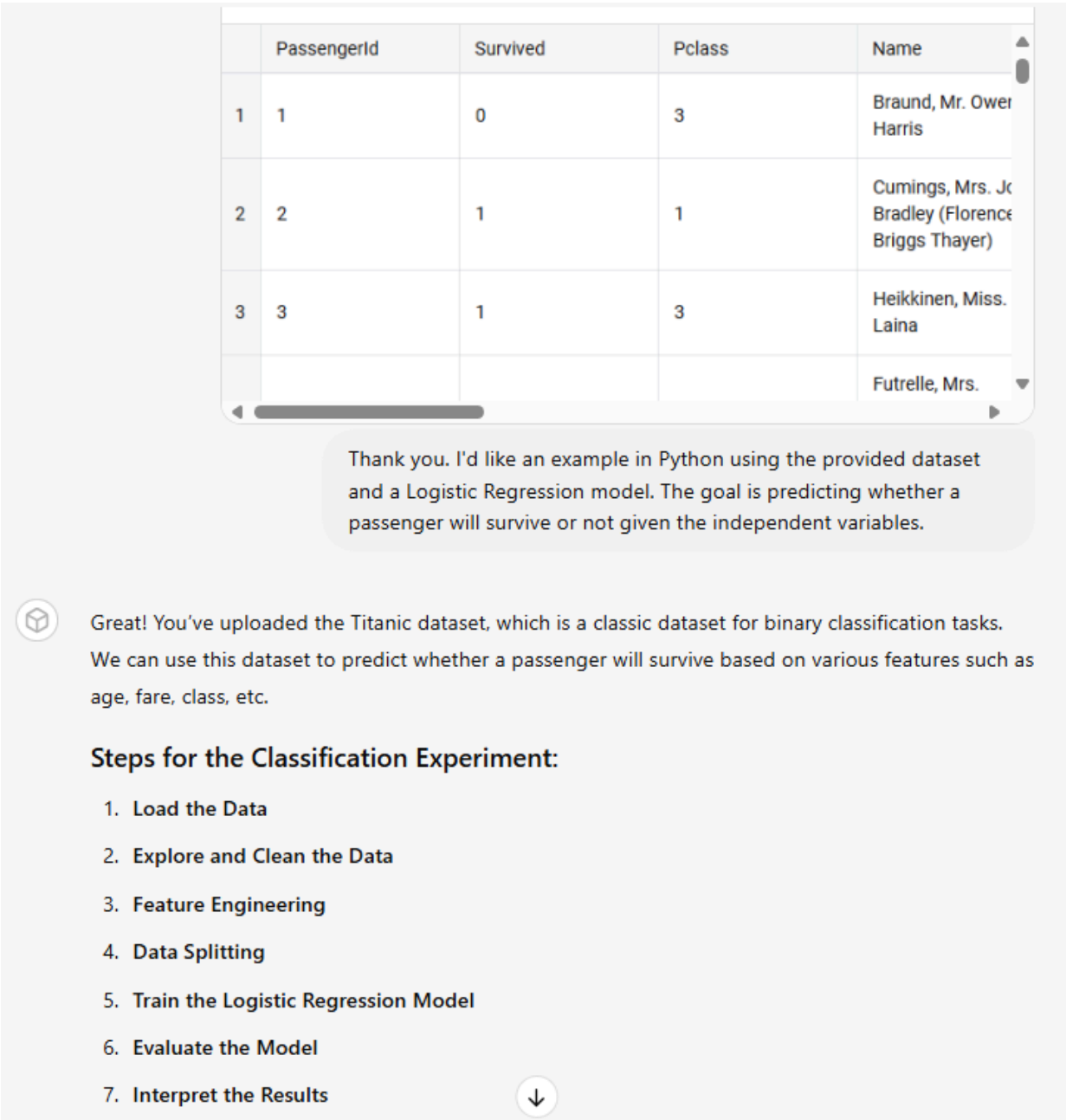
3. Choose a Model

Figure 20: Example of the DataScience Assistant providing guidance on how to tackle a classification problem.

- Following these instructions, we are now ready to set up our experiment. We want to tackle the well-known Titanic passengers' survival prediction. We will upload our dataset (you can find many free versions online, I downloaded mine at https://github.com/datasciencedojo/datasets/blob/master/titanic.csv) and leverage a Logistic Regression model.

  The Titanic survival prediction task is a classic problem in data science and machine learning. The goal is to predict whether a passenger on the Titanic would survive or not, based on various features like age, gender, passenger class, fare, and more. This task is typically used to teach classification techniques, where the model is trained on a labeled dataset and then used to predict outcomes on new data. The challenge involves selecting relevant features, handling missing data, and choosing the appropriate machine learning algorithm to achieve accurate predictions.

- This is my query:



	PassengerId	Survived	Pclass	Name
1	1	0	3	Braund, Mr. Owe Harris
2	2	1	1	Cumings, Mrs. Jo Bradley (Florence Briggs Thayer)
3	3	1	3	Heikkinen, Miss. Laina
				Futrelle, Mrs.

Thank you. I'd like an example in Python using the provided dataset and a Logistic Regression model. The goal is predicting whether a passenger will survive or not given the independent variables.

Great! You've uploaded the Titanic dataset, which is a classic dataset for binary classification tasks. We can use this dataset to predict whether a passenger will survive based on various features such as age, fare, class, etc.

Steps for the Classification Experiment:

1. **Load the Data**
2. **Explore and Clean the Data**
3. **Feature Engineering**
4. **Data Splitting**
5. **Train the Logistic Regression Model**
6. **Evaluate the Model**
7. **Interpret the Results**

Figure 21: Example of DataScience Assistant designing the Titanic survival experiment.

I will now share some screenshots of the model's response per each step:

1. Load the data:

1. Load the Data

First, we'll load the dataset and take a look at the first few rows.

I'll begin by loading and displaying the first few rows of your dataset.

The dataset contains the following columns:

- **PassengerId**: Unique identifier for each passenger.
- **Survived**: The target variable (1 = Survived, 0 = Did not survive).
- **Pclass**: Passenger class (1st, 2nd, 3rd).
- **Name**: Name of the passenger.
- **Sex**: Gender of the passenger.
- **Age**: Age of the passenger.
- **SibSp**: Number of siblings or spouses aboard the Titanic.
- **Parch**: Number of parents or children aboard the Titanic.
- **Ticket**: Ticket number.
- **Fare**: Passenger fare.
- **Cabin**: Cabin number.
- **Embarked**: Port of Embarkation (C = Cherbourg; Q = Queenstown; S = Southampton).

Figure 22: Example of DataScience Assistant executing the step 1 of the experiment.

2. Explore and clean the data:

2. Explore and Clean the Data

Next, we'll check for any missing values and handle them appropriately. We'll also drop any columns that won't be useful for the classification task.

Let's proceed with that. [>_]

The dataset has missing values in the following columns:

- **Age**: 177 missing values
- **Cabin**: 687 missing values
- **Embarked**: 2 missing values

Handling Missing Data:

1. **Cabin**: The Cabin column has a large number of missing values, so it might be best to drop this column as it could introduce noise into the model.
2. **Age**: We can fill missing values in the Age column with the median age, as it's a continuous variable.
3. **Embarked**: We can fill the missing values in the Embarked column with the most common port of embarkation.

Let's clean the dataset accordingly. [>_]

The missing values have been successfully handled. Now the dataset is clean and ready for the next steps.

Figure 23: Example of DataScience Assistant exploring and cleaning data.

Note that, when you see the symbol [>_], it means that the code interpreter plugin has been triggered. You can click on it to see the executed code:

Analysis

python

Always show details

Copy code

```python
import pandas as pd

# Load the dataset
file_path = '/mnt/data/titanic (1).csv'
titanic_df = pd.read_csv(file_path)

# Display the first few rows of the dataset
titanic_df.head()
```

Result

```
   PassengerId  Survived  Pclass  \
0            1         0       3
1            2         1       1
2            3         1       3
3            4         1       1
4            5         0       3

                                                Name     Sex
0                            Braund, Mr. Owen Harris    male
1  Cumings, Mrs. John Bradley (Florence Briggs Th...  female
2                             Heikkinen, Miss. Laina  female
3       Futrelle, Mrs. Jacques Heath (Lily May Peel)  female
4                           Allen, Mr. William Henry    male

   Parch            Ticket     Fare Cabin Embarked
0      0         A/5 21171   7.2500   NaN        S
```

Figure 24: Example of DataScience Assistant's generated code leveraging the code interpreter plugin.

1. Feature Engineering:

3. Feature Engineering

We need to prepare the features for the model:

- **Encoding Categorical Variables:** Convert categorical variables such as `Sex` and `Embarked` into numerical values.
- **Drop Irrelevant Columns:** Drop columns like `PassengerId`, `Name`, and `Ticket` that are not useful for the classification.

Let's proceed with these steps. [>_]

The dataset has been successfully transformed, and it now includes the following features:

- **Survived:** Target variable.
- **Pclass:** Passenger class.
- **Sex:** Gender (0 = male, 1 = female).
- **Age:** Age of the passenger.
- **SibSp:** Number of siblings/spouses aboard.
- **Parch:** Number of parents/children aboard.
- **Fare:** Passenger fare.
- **Embarked_Q:** Embarked at Queenstown (1 if true, 0 otherwise).
- **Embarked_S:** Embarked at Southampton (1 if true, 0 otherwise).

Figure 25: Example of DataScience Assistant doing feature engineering.

2. Data splitting:

4. Data Splitting

Next, we'll split the dataset into training and testing sets.

Let's proceed with that. [>_]

The dataset has been successfully split into training and testing sets:

- **Training set:** 623 samples
- **Testing set:** 268 samples

Figure 26: Example of DataScience Assistant splitting the dataset into training and test set.

3. Train the model:

5. Train the Logistic Regression Model

Now, we'll train a Logistic Regression model using the training data.

Let's proceed with training the model. [>_]

The Logistic Regression model has been trained, and predictions have been made on the test set. The first few predictions are as follows:

- **0:** Did not survive
- **1:** Survived

Figure 27: Example of DataScience Assistant training a Logistic Regression model.

4. Evaluate the model:

6. Evaluate the Model

Next, we'll evaluate the model's performance using common classification metrics such as accuracy, precision, recall, F1-score, and the confusion matrix.

Let's calculate these metrics. [>_]

The model's performance on the test set is as follows:

- **Accuracy**: 80.97%
- **Precision**: 79.41%
- **Recall**: 72.97%
- **F1-Score**: 76.06%

Confusion Matrix:

lua Copy code

```lua
[[136,  21],  # True Negatives, False Positives
 [ 30,  81]]  # False Negatives, True Positives
```

Interpretation:

- The model has a decent accuracy, with 80.97% of predictions being correct.
- Precision indicates that when the model predicts survival, it is correct 79.41% of the time.
- Recall suggests that the model correctly identifies 72.97% of the actual survivors.
- The F1-score, which balances precision and recall, is 76.06%.

Figure 28: Example of DataScience Assistant evaluating the model's output.

That's pretty cool! It is extremely accurate and can save a lot of time. Plus, if you think about data scientists working in large enterprises on many projects, having a similar assistant can also help in following fixed standards among projects, so that the maintenance is aligned across teams.

The last thing that we will ask our GPT is to push the code on our Repo. Let's see how it works:

Now push this code on my GitHub repo

Please provide the following details so I can push the code to your GitHub repository:

1. **Repository owner:** The username of the owner of the repository.
2. **Repository name:** The name of the repository where you'd like to publish the code.
3. **File path:** The path where you'd like to save the file in the repository (e.g., `scripts/titanic_logistic_regression.py` or simply `titanic_logistic_regression.py`).
4. **Commit message:** A short description of the change you're making.
5. **Branch:** The branch where you'd like to create the file (e.g., `main`).

Valentina-Alto, GPTActions, titanic_logistic_regression.py, "upload with gpt action"

> [debug] Calling HTTP endpoint
> [debug] Calling HTTP endpoint
> [debug] Response received

Talked to api.github.com

The Python code for logistic regression on the Titanic dataset has been successfully uploaded to your GitHub repository. You can view the file here.

If you need further assistance, feel free to ask!

Figure 29 Example of DataScience Assistant leveraging the Action to push the code on GitHub.

If we click on the link, we can see that the file has been successfully uploaded:

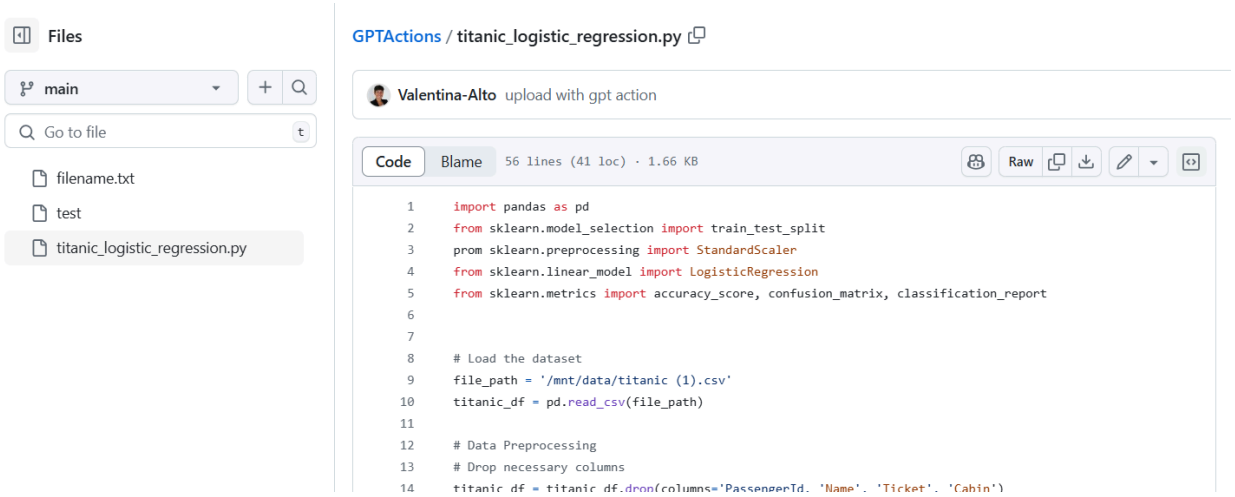


Figure 30: File uploaded via DataScience Assistant Action.

And it worked! Again, this is an example of how GPTs can speed up developers and data scientists' productivity. By designing data science experiments following a common framework and enabling push workflows without the need of switching to GitHub can save previous time, so that data scientists can focus on the core aspects of their projects.

Marketing Assistant

As we saw in Chapter 6 – Mastering Marketing with ChatGPT, AI assistant can be extremely valuable when it comes to this domain. In fact, generating text content – like social media posts, blog articles or marketing campaigns – is probably the activity in which these models perform at their best.

> A copywriter is a professional who specializes in writing persuasive and engaging content, often for marketing and advertising purposes. Their work typically includes writing promotional materials such as advertisements, brochures, websites, emails, social media posts, and other forms of content designed to persuade an audience to take a specific action, like making a purchase or subscribing to a service.

In this section, we are going to create a Copywrite assistant which is tailored to this kind of activity. To do so, I named my assistant Copywrite Companion, and I set the following configuration components:

- Instructions:

Copywrite Companion is a versatile assistant designed to help users with various writing tasks. It specializes in generating product sheets based on user input, including text and images, crafting email campaigns for newsletters or promotional purposes, creating engaging visuals for new products, and writing tailored social media posts for different platforms. It ensures content is persuasive, engaging, and aligned with the target audience, aiming to increase engagement and promote products or services effectively. The assistant adapts its style to the user's needs and strives for creativity, clarity, and relevance in all outputs. It communicates in a casual, friendly tone, making interactions feel approachable and easygoing.

- Conversation starters:

Can you write a product description for my new product?

I need a catchy headline for an article about climbing

Give me a few ideas for a blog post about running

- Capabilities:

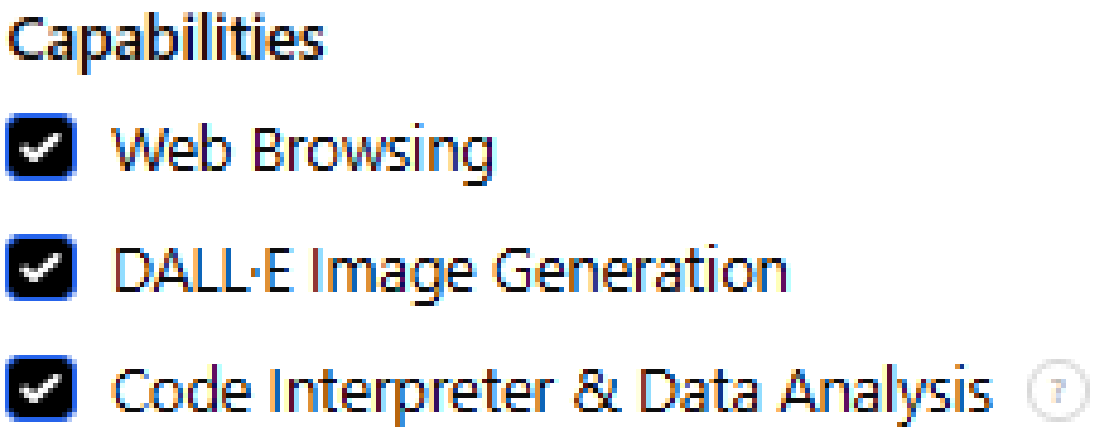


Figure 31: Copywrite Companion's enabled plugins.

Let's see it in action:

- I'll first ask it to write a product sheet starting from a picture provided (output truncated):

generate a product sheet for these running shoes

Product Sheet: VelocityX Running Shoes

Product Name: VelocityX Running Shoes

Category: Men's Athletic Footwear

Key Features:

- **Advanced Sole Technology:** Equipped with the latest Flexi-Grip sole that adapts to your stride, offering unparalleled traction and stability on any surface. The responsive cushioning ensures maximum energy return with every step, keeping you light on your feet.
- **Breathable Design:** The VelocityX shoes feature a high-performance, moisture-wicking mesh upper that keeps your feet cool and dry, even during the most intense workouts. The breathable material also reduces odor and ensures a comfortable fit throughout your run.
- **Lightweight Construction:** Built with ultra-lightweight materials, these shoes won't weigh you down. The streamlined design is perfect f unners seeking both speed and durability without

Figure 32: Example of Copywrite Companion generating a product description.

- As copywriter, we might want to insert this set of information into more structured repository, like an excel file. Let's ask the assistant to do so, leveraging its code interpreter plugin:

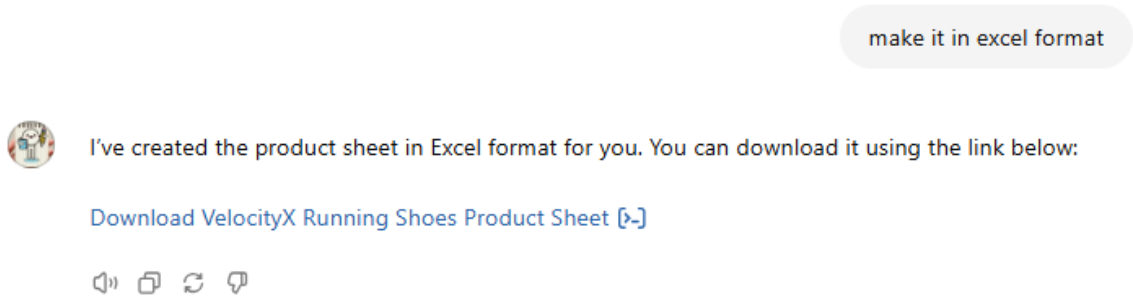
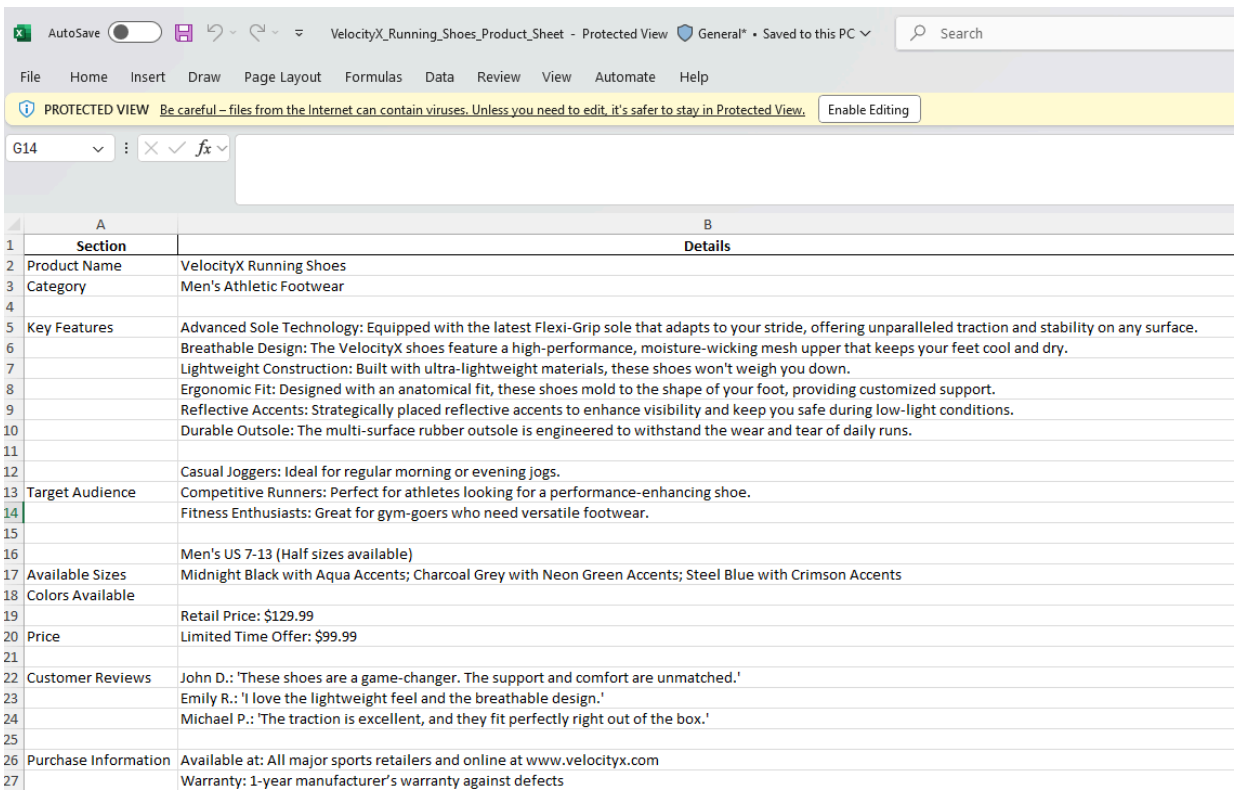


Figure 33: Example of Copywrite Companion converting its previous response into an excel file, leveraging the code interpreter plugin.

- And this is the final result:



	A	B
1	**Section**	**Details**
2	Product Name	VelocityX Running Shoes
3	Category	Men's Athletic Footwear
4		
5	Key Features	Advanced Sole Technology: Equipped with the latest Flexi-Grip sole that adapts to your stride, offering unparalleled traction and stability on any surface.
6		Breathable Design: The VelocityX shoes feature a high-performance, moisture-wicking mesh upper that keeps your feet cool and dry.
7		Lightweight Construction: Built with ultra-lightweight materials, these shoes won't weigh you down.
8		Ergonomic Fit: Designed with an anatomical fit, these shoes mold to the shape of your foot, providing customized support.
9		Reflective Accents: Strategically placed reflective accents to enhance visibility and keep you safe during low-light conditions.
10		Durable Outsole: The multi-surface rubber outsole is engineered to withstand the wear and tear of daily runs.
11		
12		Casual Joggers: Ideal for regular morning or evening jogs.
13	Target Audience	Competitive Runners: Perfect for athletes looking for a performance-enhancing shoe.
14		Fitness Enthusiasts: Great for gym-goers who need versatile footwear.
15		
16		Men's US 7-13 (Half sizes available)
17	Available Sizes	Midnight Black with Aqua Accents; Charcoal Grey with Neon Green Accents; Steel Blue with Crimson Accents
18	Colors Available	
19		Retail Price: $129.99
20	Price	Limited Time Offer: $99.99
21		
22	Customer Reviews	John D.: 'These shoes are a game-changer. The support and comfort are unmatched.'
23		Emily R.: 'I love the lightweight feel and the breathable design.'
24		Michael P.: 'The traction is excellent, and they fit perfectly right out of the box.'
25		
26	Purchase Information	Available at: All major sports retailers and online at www.velocityx.com
27		Warranty: 1-year manufacturer's warranty against defects

Figure 34: Excel file generated by Copywrite Companion

Let's now ask the assistant to generate an Instagram post to sponsor our shoes:

now create an Instagram post about these shoes, including a picture

Here's an Instagram post idea for the VelocityX Running Shoes:

Caption:

🌟 **Unleash Your Full Potential with VelocityX!** 🌟

Ready to elevate your run? Meet the VelocityX Running Shoes—where style meets performance. 🏃‍♂️💨

🔥 **Key Features:**

- Advanced Flexi-Grip sole for unbeatable traction.
- Breathable mesh upper to keep you cool.
- Lightweight design for speed without compromise.
- Reflective accents for those early morning or late-night runs.

Whether you're hitting the track or the treadmill, VelocityX has got your back (and your feet!). Don't just run—*run with purpose*.

💥 **Limited Time Offer:** Get yours now for just $99.99! (Retail: $129.99)

👉 Shop now: www.velocityx.com

#VelocityX #RunningShoes #RunWithPurpose #FitnessGoals #MensAthleticShoes #RunnersLife #NewRelease

---

Image Description:

A dynamic shot of a runner mid-stride, showcasing the sleek design of the VelocityX Running Shoes. The image captures the shoe's details, including the advanced sole and breathable mesh upper, against the backdrop of a sunlit running track. The runner is in motion, emphasizing the lightweight and ergonomic design, with the shoes' reflective accents subtly glinting in the light.

Would you like me to generate the image for this post?

Figure 35: Example of Copywrite Companion generating an Instagram post.

As you can see, the assistant also proposed an image description to leverage the DALL-E plugin. Since the description makes sense to me, I'll go ahead and ask it to generate it:

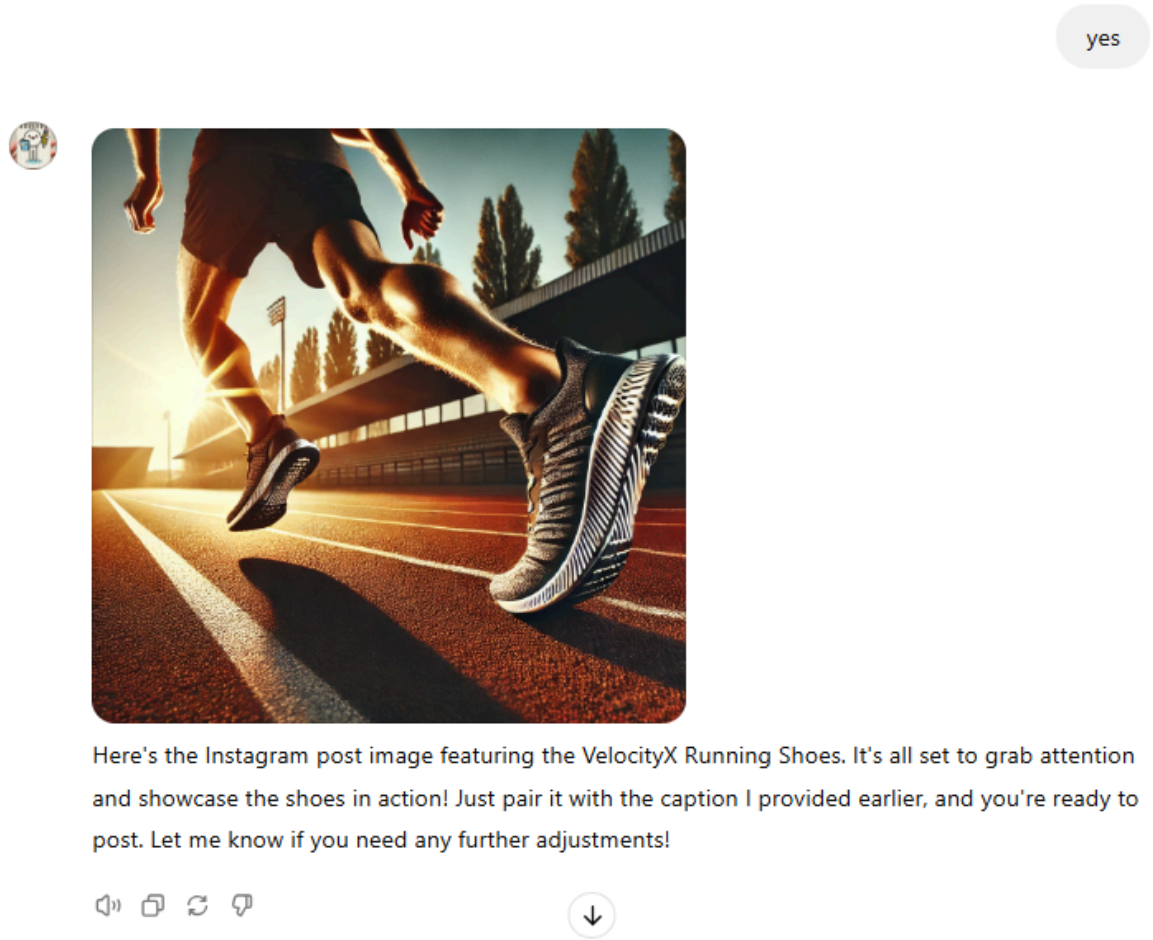


Figure 36 Example of Copywrite Companion generating an image leveraging the DALL-E plugin.

Finally, I want to have an idea of how big competitor brands – like Nike and Adidas – are doing their marketing activities. To do so, I'll ask my assistant to gather some evidence from the web:

Can you tell me whether Nike and Adidas are now sponsoring similar running shoes? If yes, analyze their marketing activities and tell me which style they are using

Searched 4 sites

Yes, both Nike and Adidas are actively sponsoring running shoes similar to the VelocityX, and their marketing strategies are notably distinct yet impactful.

Nike:
Nike continues to focus on its people-centric and values-based marketing approach. Their strategy often leans heavily on storytelling that ties into broader social and cultural narratives. For instance, Nike's 2018 "Dream Crazy" campaign featuring Colin Kaepernick is a prime example of their willingness to take risks by aligning with socially charged issues, a move that bolstered their brand value by $6 billion. This approach resonates particularly well with younger generations, such as Gen Z, who value authenticity and social responsibility in brands.

Nike's sponsorships often revolve around high-profile athletes and major sports events, creating campaigns that blend athletic achievement with cultural relevance. They emphasize technological innovation in their products, such as the use of their proprietary Nike Air and React foams, which are designed to enhance performance while maintaining comfort and support. Nike also invests heavily in creating visually stunning and emotionally resonant advertisements that elevate the perceived value of their products (Marketing Dive) (RunToTheFinish).

Adidas:
Adidas, on the other hand, blends athletic performance with a strong focus on lifestyle and environmental responsibility. They have built a brand identity around sustainability, frequently promoting products made with recycled materials and eco-friendly technologies, such as their Boost foam and Primegreen materials. Adidas positions itself not just as a sportswear brand but as a cultural icon, often blurring the lines between sports performance and streetwear fashion.

Their sponsorships are more diversified, covering not only athletes but also musicians, artists, and other cultural influencers. This strategy helps Adidas connect with a broader audience beyond just sports enthusiasts. Additionally, their marketing campaigns often emphasize unity, environmental consciousness, and inclusivity, which are key values that resonate well with today's socially aware consumers (Marketing Dive) (Adidas).

In summary, while Nike's marketing is more about leveraging cultural moments and social issues through powerful storytelling and athlete endorsements, Adidas focuses on sustainability, lifestyle integration, and a broader cultural appeal. Both styles are effective but cater to slightly different aspects of consumer identity and values.

aspects of consumer identity and values.

Figure 37: Example of Copywrite Companion doing a competitive analysis leveraging the web browsing plugin.

As you can see, our companion correctly leveraged the web browsing plugin to retrieve the required information. Plus, it gave us powerful insight into which leverages the two competitor companies are investing, so that we might think (or ask our companion) about unique differentiators that could let our brand outstand in a competitive market.

We could also do a step ahead and ask for more specific insights about the competition by leveraging the code interpreter and data analysis plugin. Let's say, for example, that we gathered an excel sheet with the following structure:

Year	Company	Revenue (Billions USD)	Market Share (%)	Advertising Spend (Millions USD)	Social Media Engagement (Millions)	Brand Value (Billions USD)
2021	Competitor A	44,5	25	3	150	34
2021	Competitor B	22	15	2	120	16
2022	Competitor A	46	26	3,2	160	35,5

Figure 38: Competitive analysis on an Excel sheet

And now we want to generate some visuals out of it. Let's ask our Copywriter Companion to do so:

	Year	Company	Revenue (Billions USD)	Market Share (%)
1	2021	Competitor A	44.5	25
2	2021	Competitor B	22	15
3	2022	Competitor A	46	26
4	2022	Competitor B	23.5	16
5	2023	Competitor A	48	27
6	2023	Competitor B	25	17

Generate relevant graphs from this file. Plus, generate an executive report about cost and benefits of setting a more competitive price for our product.

Revenue Trend Over The Years

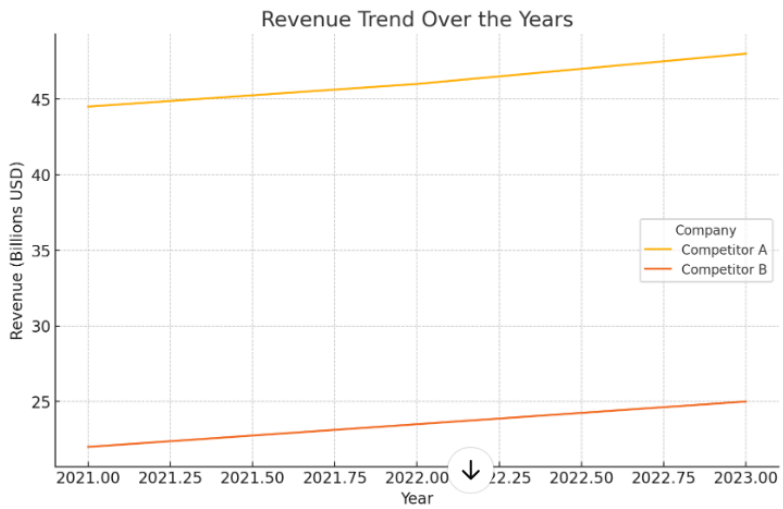
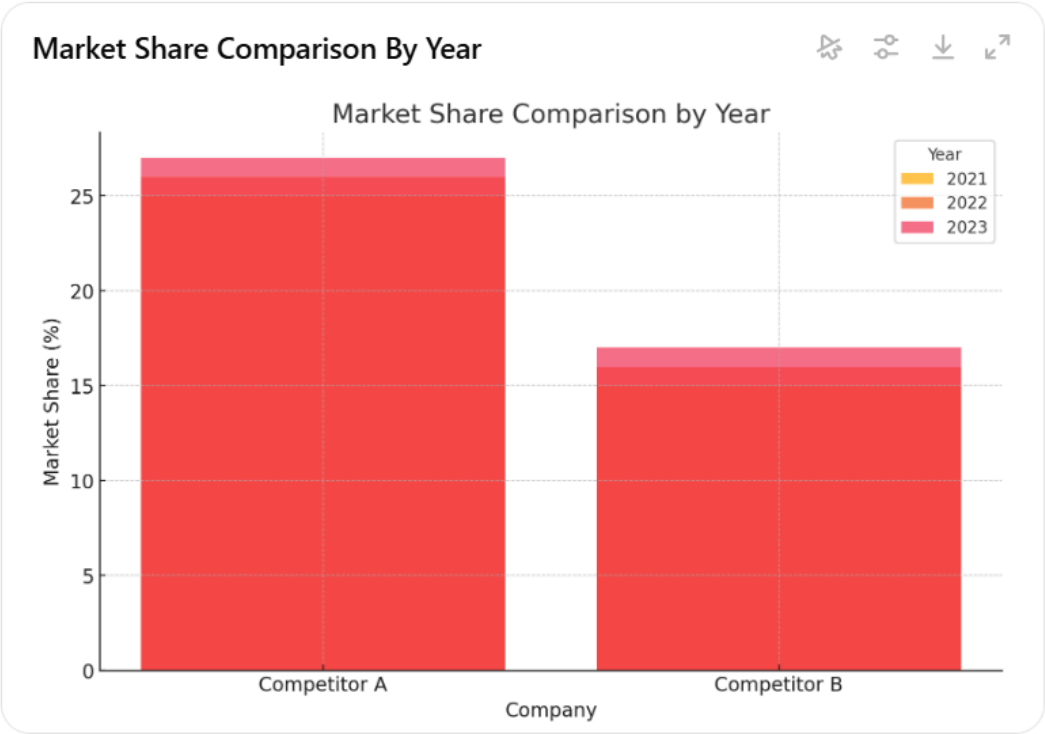


Figure 39: Example of bar chart and scatter plot generated by Copywrite companion.



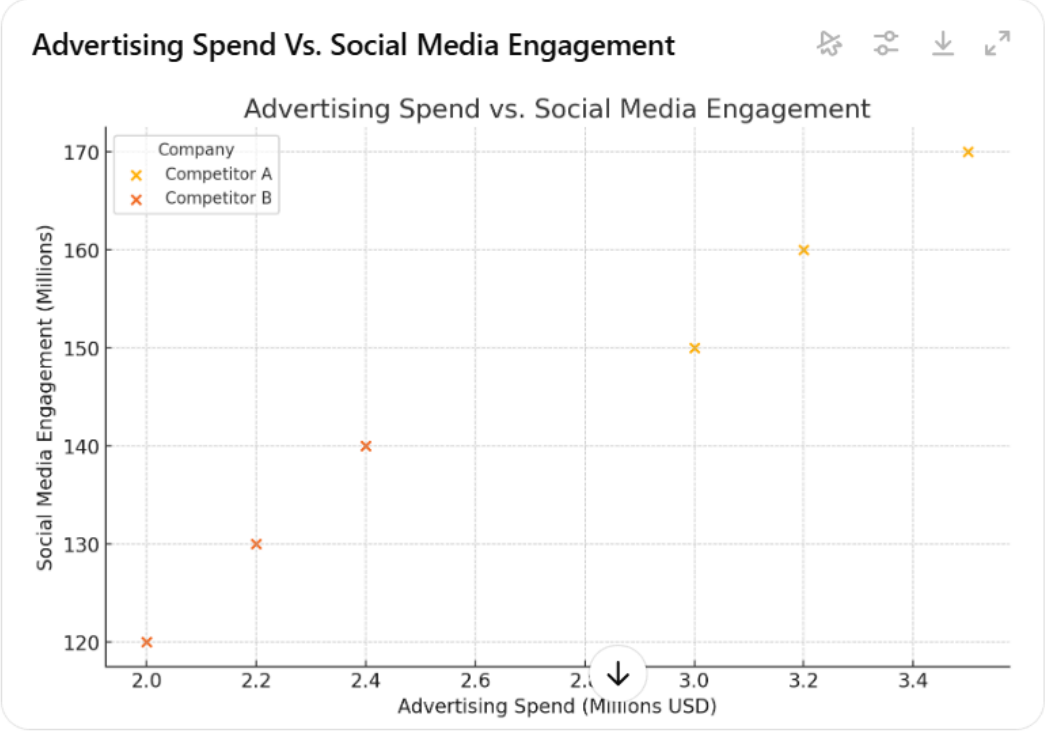


Figure 40: Example of line chart generated by Copywrite companion.

Including, as requested, the executive report:

Executive Report: Cost and Benefits of Setting a More Competitive Price

Objective: To assess the potential cost and benefits of setting a more competitive price for our product, using the available data on competitors' revenue, market share, advertising spend, and brand value.

Cost Analysis:

- **Reduced Revenue Per Unit**: Setting a more competitive price will likely decrease the revenue per unit sold. However, this could be offset by an increase in sales volume.
- **Increased Advertising Spend**: To maintain or grow market share while reducing prices, an increase in advertising spend may be necessary, as seen in the correlation between advertising spend and social media engagement.
- **Brand Value**: Aggressive price reductions might affect brand perception, potentially lowering brand value if not managed carefully.

Benefit Analysis:

- **Market Share Growth**: Competitor A's strategy of sustained revenue growth is accompanied by significant market share gains. A competitive pricing strategy could help increase our market share, potentially positioning us closer to or ahead of major competitors.
- **Higher Sales Volume**: Lower prices could lead to higher sales volumes, which might compensate for reduced per-unit revenue.
- **Enhanced Social Media Engagement**: With proper advertising spend, competitive pricing could enhance social media engagement, further driving brand visibility and customer acquisition.

Strategic Considerations:

- **Balance Price and Perceived Value**: It's crucial to maintain a balance between competitive pricing and the perceived value of our product to avoid eroding brand equity.
- **Monitor Competitor Reactions**: Competitors may respond with their own pricing adjustments, which could trigger a price war. A careful assessment of the market's elasticity and our cost structure is necessary to ensure sustainable profitability.
- **Optimize Advertising Strategy**: As indicated by the positive correlation between advertising spend and social media engagement, a targeted advertising strategy should be implemented to maximize the impact of the price reduction.

Figure 41: Example of an Executive report generated by Copywrite companion.

Overall, tailoring ChatGPT for marketing activities can be extremely useful when it comes to generating new content, designing marketing strategies and doing competitive analysis on the

web.

Research Assistant

In this scenario, we are going focus on research once more, yet this time with a particular focus on papers retrieval. More specifically, we want our assistant to be able to do the following:

- Retrieving information from custom knowledge base that we provide
- Integrating the custom documents with papers only from Arxiv, enabling the web plugin for this task
- Retrieving from a Database (in our case, it will be hosted in Notion) existing ongoing work from other researchers, so that we don't risk elaborating an essay which is already covered by someone else.

Let's see all the steps.

1. Upload custom documents. For this purpose, I'll use two papers about Image Classification in Machine Learning: "*CIFAKE: image classification and explainable identification of ai-generated synthetic images*" from J. Bird et al., and "*A Comprehensive Study of Vision Transformers in Image Classification Tasks*" from Khalis et al.

You can upload them in the proper section in the configuration pane:

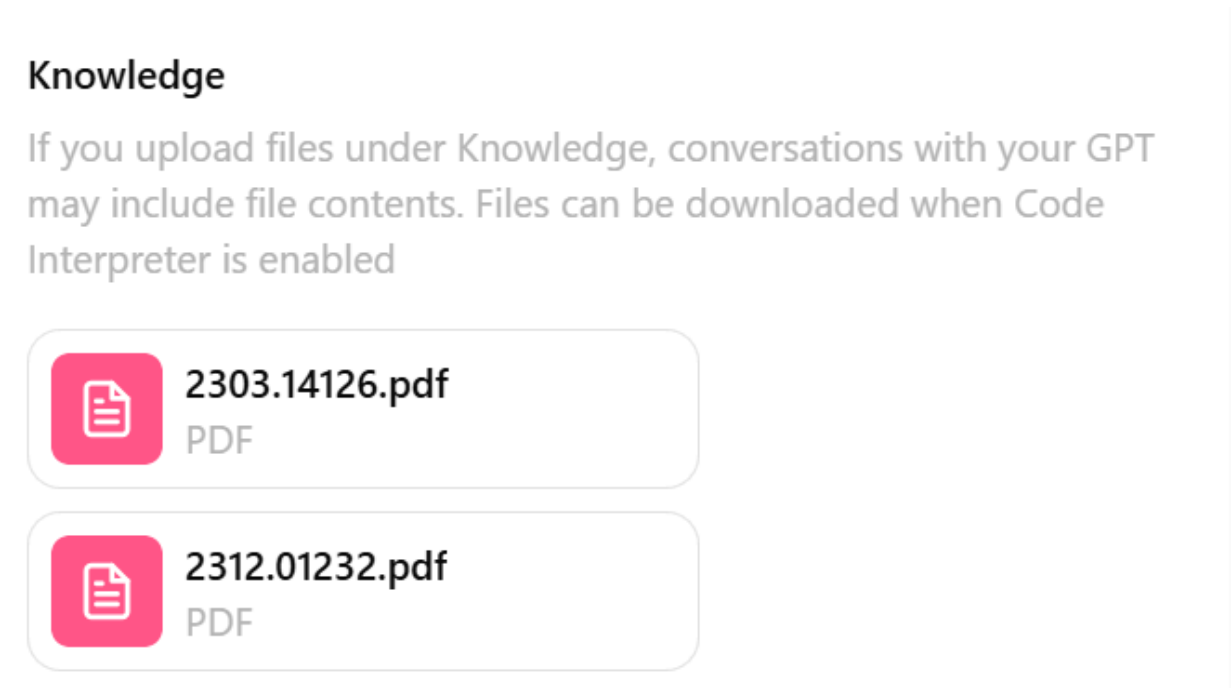


Figure 42: Uploading custom documents in the configuration pane

1. Integrating custom documents with web references. To do so, we need to enable the web plugin:

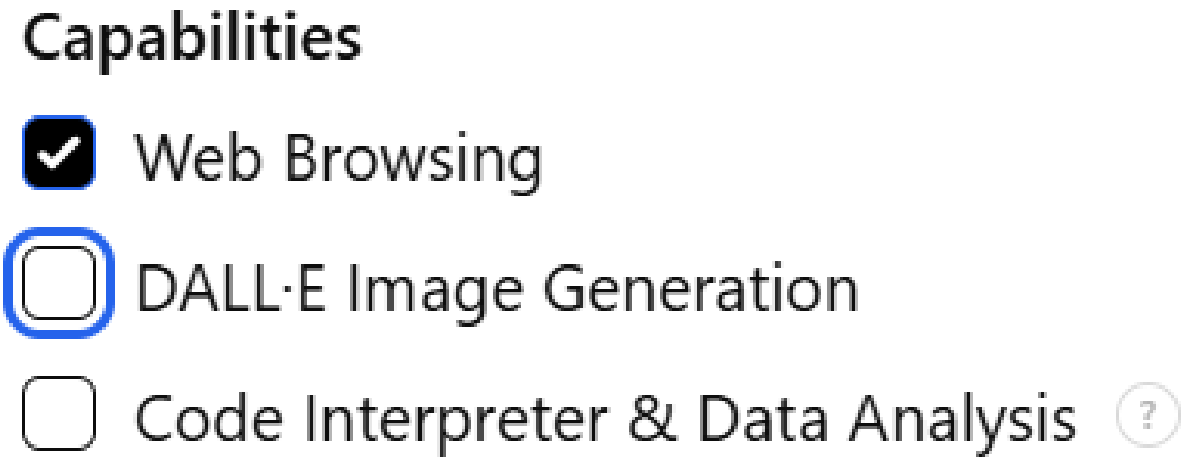


Figure 43: Enabling the web browsing plugin in the configuration pane

Plus, we also need to specify the assistant to only navigate through the Arxiv archive. We will see how to specify that in the set of instructions we will create.

1. Retrieve information from a Notion Database. Here, the idea is that, as researchers, we might come up with ideas that are already being studied and developed by other fellow colleagues. Imagine that we keep track of all the ongoing research in a Notion Database that has the following structure:

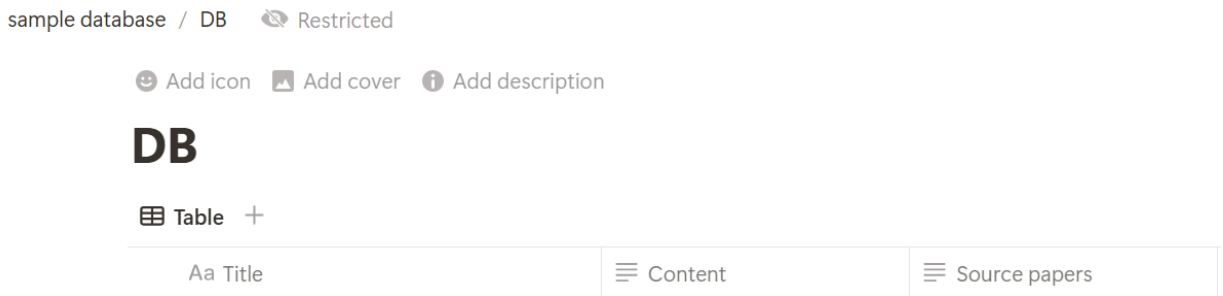


Figure 44: Notion Database structure.

To do so, we need to create a GPT actions. To do so, there are two steps to follow:

1. In your Notion workspace, you need to create a new connection marked as internal, along with a new Internal Integration Secret (or API key) will be created. You can call this connection “chatgpt” or similar.
2. In your GPT configuration pane, you need to set a new Action schema. Since in our case we need to query a specific Database, the schema will look like the following:
3. When it comes to authentication, you can click on "Authentication" and choose "API Key". Enter in the information below.
    - API Key: Use Internal Integration Secret from your newly created connection in Notion

- Auth Type: Bearer

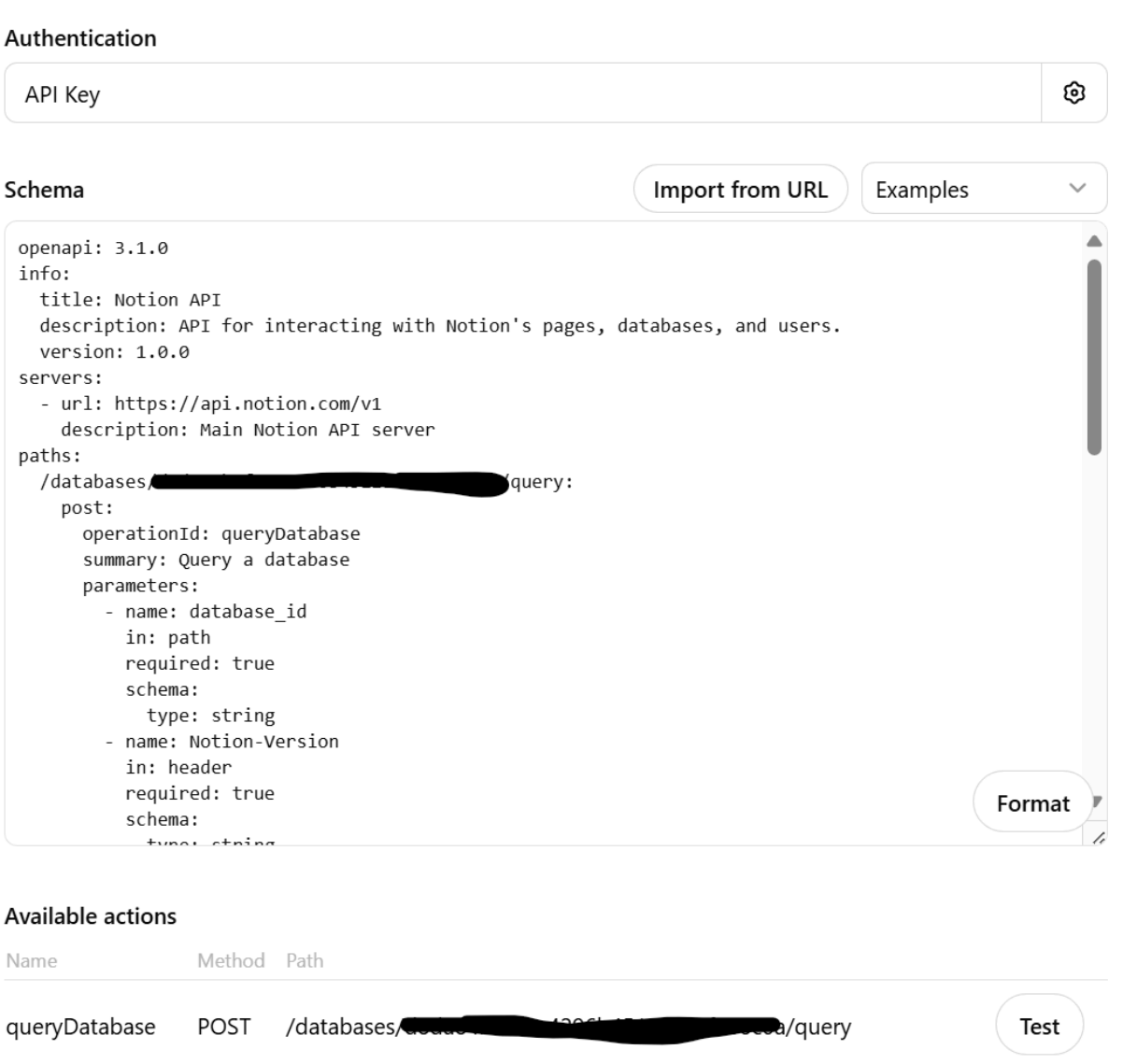


Figure 45: Notion Action Schema.

This is how the whole schema looks like:

```
openapi: 3.1.0
info:
  title: Notion API
  description: API for interacting with Notion's pages, databases, and users.
  version: 1.0.0
servers:
  - url: https://api.notion.com/v1
    description: Main Notion API server
paths:
  /databases/{database_id}/query:
    post:
      operationId: queryDatabase
```

```
      summary: Query a database
      parameters:
        - name: database_id
          in: path
          required: true
          schema:
            type: string
        - name: Notion-Version
          in: header
          required: true
          schema:
            type: string
          example: 2022-06-28
          constant: 2022-06-28
      requestBody:
        required: true
        content:
          application/json:
            schema:
              type: object
              properties:
                filter:
                  type: object
                sorts:
                  type: array
                  items:
                    type: object
                start_cursor:
                  type: string
                page_size:
                  type: integer
      responses:
        '200':
          description: Successful response
          content:
            application/json:
              schema:
                type: object
                properties:
                  object:
                    type: string
                  results:
                    type: array
                    items:
                      type: object
                  next_cursor:
                    type: string
                  has_more:
                    type: boolean
```

Great, now that we have all our ingredients, we need to set our system message and, optionally, our conversation starters. In this scenario, I set the following instructions:

You are an AI research assistant tasked with helping researchers by leveraging a variety of tools and resources. Your primary responsibilities include retrieving, integrating, and cross-referencing information from custom knowledge bases, academic papers on Arxiv, and ongoing research projects

stored in a Notion database. Follow these guidelines to ensure your assistance is accurate, comprehensive, and avoids redundancy:

1. *Always answer based on the provided documents*
2. *If you feel it is needed, extend your search on the web. the ONLY ONE SITE you can navigate through is Arxiv.*
3. *If asked by the user, check whether the topic is already covered in the Notion DB.*

And the following conversation starters:

1. *How do VGGNet, ResNet, and Inception differ in accuracy and efficiency for image classification?*
2. *What challenges do image classification models face, and how do techniques like data augmentation and transfer learning help?*
3. *How does CNN depth affect image classification, and how do ResNet's residual connections help?*

The final product looks like so:

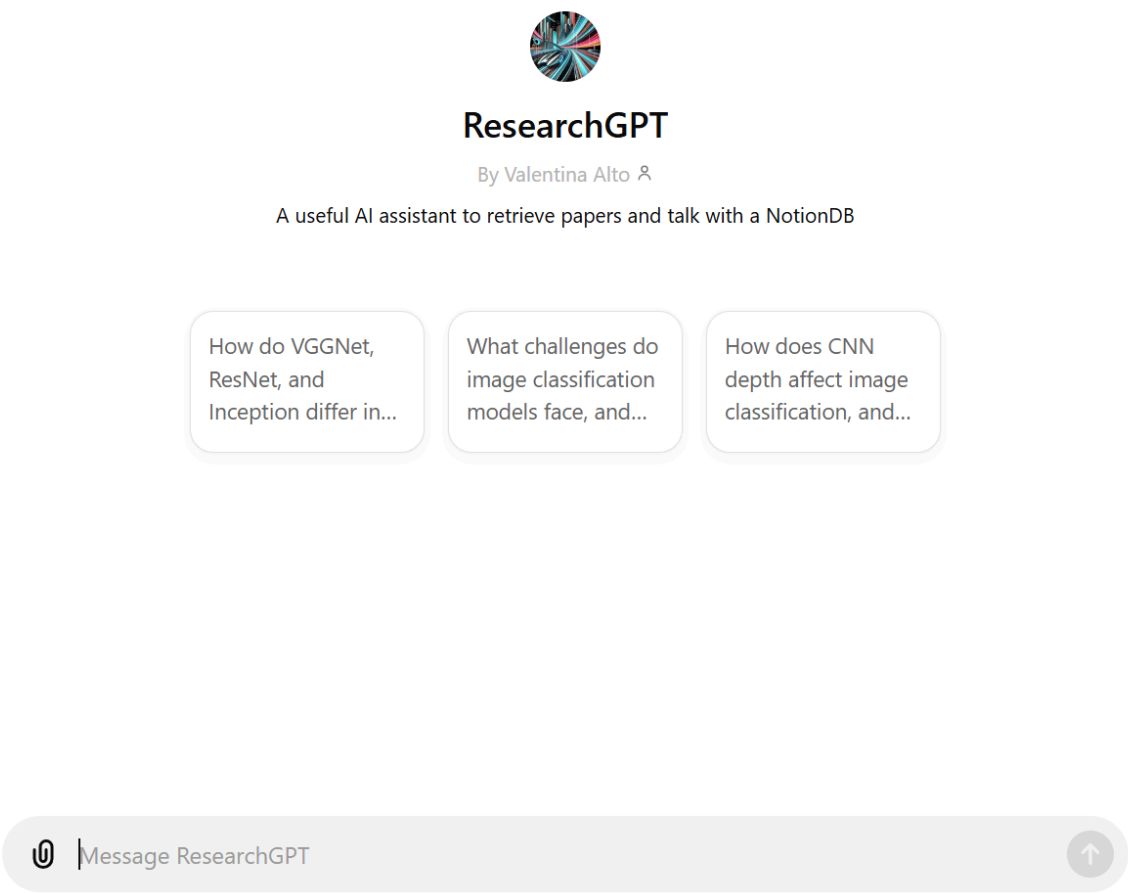


Figure 46: Landing page of our ResearchGPT

Let's test it:

1. I'll first ask a generic question, that will be addressed with the provided papers:

How do VGGNet, ResNet, and Inception differ in accuracy and efficiency for image classification?

VGGNet, ResNet, and Inception: Differences in Accuracy and Efficiency for Image Classification

1. **VGGNet**

- **Architecture:** VGGNet is characterized by its deep, sequential layers of 3x3 convolutional filters. The design is simple, with a uniform architecture where each layer is similar except for the number of filters.
- **Accuracy:** VGGNet achieves high accuracy on image classification tasks due to its depth, which allows it to learn complex features.
- **Efficiency:** VGGNet is less efficient compared to more modern architectures like ResNet and Inception. Its large number of parameters (due to the dense, sequential layers) leads to higher computational costs and more memory usage.

Figure 47: Example of ResearchGPT retrieving knowledge from provided documents

2. Then, I want to see whether the topic is already covered:

is this topic already covered in the database?

Talked to api.notion.com

Yes, the topic has already been covered in the database under the title "Deep Learning for Image Classification: A Comparative Study." This entry compares the performance of models like VGGNet, ResNet, and Inception on standard datasets such as CIFAR-10 and ImageNet, noting that ResNet provides superior accuracy in complex image recognition tasks.

Figure 48: Example of ResearchGPT talking to Notion with the pre-defined action.

3. Finally, I want to integrate the topic to make it unique:

I want to make this topic unique. Suggest me other 3 arxiv papers to integrate my topic with

3/3

To make your topic unique, you can integrate it with insights from the following three arXiv papers:

1. **"EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks"**
   - **Paper ID:** 1905.11946
   - **Summary:** This paper introduces EfficientNet, a model that optimizes the balance between network depth, width, and resolution. Integrating your comparison with EfficientNet can provide a modern contrast, focusing on how scaling techniques affect accuracy and efficiency relative to VGGNet, ResNet, and Inception.
   - **Link:** arXiv:1905.11946
2. **"MixConv: Mixed Depthwise Convolutional Kernels"**
   - **Paper ID:** 1907.09595
   - **Summary:** MixConv proposes mixed depthwise convolutional kernels, allowing models to extract features at different scales more efficiently. Incorporating this into your study can add a layer of complexity regarding how various convolutional techniques affect performance and efficiency.
   - **Link:** arXiv:1907.09595
3. **"Xception: Deep Learning with Depthwise Separable Convolutions"**
   - **Paper ID:** 1610.02357
   - **Summary:** Xception explores depthwise separable convolutions, offering an efficient alternative to Inception modules. Comparing this approach with the traditional architectures could highlight efficiency improvements in modern architectures.
   - **Link:** arXiv:1610.02357

Figure 49: Example of ResearchGPT using the web browser plugin.

As you can see, our Assistant could leverage all the tools we provided it with, calling the Notion DB when needed.

Summary

In this Chapter, we explored how to get customized GPTs to address your specific goals. The possibility of tailoring ChatGPT opens a new landscape of scenarios, where highly specialized AI assistant becomes the everyday companion of professionals. Plus, OpenAI offered a no-code UI to

create GPTs, so that all citizen developers can benefit not only from the extended marketplace of available solutions, but also from their own creations.

With the plugins extensibility and the custom knowledge base, custom GPTs can serve you for limitless tasks. Then, with the addition of powerful actions, they can also communicate with the surrounding environment and evolve from “mere” generation to automation.

With this Chapter, we also conclude Part 2 of this book, where we focused on practical applications of ChatGPT. Starting from next Chapter, we are going to cover more in details how large enterprises can leverage OpenAI models and embed them into their business processes. We will cover trending enterprise use cases and see practical implementation on how to embed OpenAI’s LLMs via REST API in your applications.

References

- https://arxiv.org/abs/2312.11914
- https://arxiv.org/abs/2101.07714
- https://arxiv.org/abs/2101.07714
- https://psycnet.apa.org/record/2014-07087-000
- https://www.tandfonline.com/doi/abs/10.2753/MIS0742-1222240109
- https://www.youtube.com/watch?v=TUmyygCMMGA
- https://arxiv.org/pdf/2312.01232
- https://arxiv.org/pdf/2303.14126