Member-only story

Prompt Engineering: Custom Prompts and Map Reduce to create Summarization App with LangChain and OpenAI

Atef Ataya · Follow

8 min read · Aug 14

Listen Share More

In this article, we will create two custom prompts to replace the default prompt function in Map Reduce Chain; the map function that summarizes each chunk and the

reduce function, which combines the summarizes of the map function and generate a final summary.

Introduction

In my previous article about Stuff Documents Chain, we discussed that most LLMs have a context length or daily token limit for free. Stuff Documents Chain will not work for large documents because it will result in a prompt that is larger than the context length since it makes one call to the LLMs, meaning you need to pay to get more tokens.

Map Reduce Chain is the proper chain to use when we have larger documents since it will split the document into small chunks that fit within the token limit of the model. It will first summarize each chunk and then get a summary of the summaries. Map Reduce chain uses two prompts: one initial to summarize each chunk of data and another prompt to combine each summary into the final one. The first prompt used to summarize each chunk is called the map prompt, and the second prompt summarizes the summaries of the first prompt is called reduce function.

Here is an example of how Map Reduce is used to process a document in LangChain:

1. The document is divided into smaller chunks.
2. The map function is applied to each chunk. In this case, the map function could be a function that extracts the keywords from each chunk.
3. The output from the map function is combined with the reduce function. In this case, the reduce function could be a function that counts the number of times each keyword appears in the document.
4. The final result is the number of times each keyword appears in the data.

Here are some of the benefits of using the Map Reduce Chain in LanChain:

1. Map Reduce chain is a very efficient way to process large documents. It can be parallelized to run on multiple cores of machines, which can significantly speed up the processing time.
2. It can be used to process any data. It is not limited to text documents.
3. Map Reduce can be scaled to handle very large documents. It can be used to process terabytes or even petabytes of data.

Prompt Engineering

Prompt engineering is the process of structuring sentences so that they can be interpreted and understood by a generative AI model is such a way that its output is in accord with the user's intention. A prompt can be a description of a desired output, such as "high-quality photo of a doctor riding a horse," a command, such as "write a limerick about a cat," or a question, such as "what is the software best practices while developing an application using c#."

Understanding prompts, also called in-context learning, is an emergent ability of large language models(LLMs). LLMs are trained on massive datasets of text and code, and they can learn to generate text, translate languages, write different kinds of creative content, and answer your questions in an informative way. However, LLMs are imperfect and can sometimes produce outputs that are not in line with the user's intentions. This is where prompt engineering comes in.

Prompt engineers use their knowledge of LLMs to design prompts that are more likely to produce desired outputs. They may do this by providing more context, adding examples, or using specific keywords. For example, if you want to prompt an LLM to generate a poem about love, you might start with the prompt, "Write a poem about love, using the following words: heart, soul, and passion." This prompt provides the LLM with some context and guidance, which can help it to generate a more accurate and creative output.

Prompt Engineering best practices

Here are some best practices for writing a prompt for a large language model (LLM):

1. Be clear and concise. The prompt should be easy to understand and should be a manageable length.
2. Provide context. The prompt should give the LLM some context about the desired output. This can be done by providing examples, setting a tone, or providing a specific format.
3. Use specific keywords. The prompt should use specific keywords that the LLM is likely to understand. This can help the LLM generate a more accurate and creative output.
4. Be open-ended. The prompt should be open-ended enough for the LLM to generate various outputs. This can help to ensure that the output is creative and interesting.
5. Be patient. Finding the right prompt for the LLM may take some trial and error. Be patient and keep experimenting until you get the desired output.
6. Use examples. Please provide examples of the desired output. This can help the LLM to understand what you are looking for.
7. Set a tone. The prompt can set a tone for the desired output. For example, if you want the LLM to generate a poem, you might set a tone of sadness or joy.
8. Provide a format. You can provide the format in the prompt if you want the LLM to generate a specific output format, such as a poem or a code snippet.
9. Be creative. Don't be afraid to experiment with different prompts. The more creative you are, the more likely you are to generate interesting and unexpected outputs.

Example of prompt engineering

Best practice

> *"Write a code snipped that will print the Fibonacci sequence up to the 10th term."*

This prompt is clear and concise and provides content to the LLM. The keywords "Fibonacci" and "10th term" give the LLM a good idea of what the code snippet should

do. The prompt is also specific, which helps the LLM to generate a correct and efficient code snippet.

Weak practice

> *"Write a code snippet that prints a sequence of numbers."*

This prompt is not as clear or concise as the best practice prompt. It does not provide enough context to the LLM, so the LLM may not understand what the code snippet should do. The prompt is also not specific, which could lead to the LLM generating an incorrect or inefficient code snippet.

Building a summarising application using LangChain and OpenAI with custom prompts

Please check my previous article about map-reduce in the link below as I am going to start where we left and not from the beginning:

In the application that we created using the map_reduce chain, if we want to check the default map function, we have to access the template string from the PromtTemplate object using the following code:

```
chain.llm_chain.prompt.template
```

This is the prompt that is used to summarize each chunk.

And if we want to check the default reduce function, we can use the following code:

```
chain.combine_document_chain.llm_chain.prompt.template
```

This will access the prompt template used by the CombineDocumentsChain inside a LangChain summarization workflow.

Map Prompt

First, we have to engineer a custom prompt to replace the default prompt using the following code:

```
map_custom_prompt='''
Summarize the following text in a clear and concise way:
TEXT:`{text}`
Brief Summary:
'''
```

This prompt is clear, concise, and provides context to the LLM. The keywords “clear,” “concise,” and “Brief Summary” give the LLM a good idea of what the summary should look like.

PromptTemplate

```
map_prompt_template = PromptTemplate (
    input_variables=['text'],
    template=map_custom_prompt
)
```

So, we will create a PromptTemplate Object and initialize it with two arguments: the input variables and the template string. The input variables are the variables that will be used to fill in the placeholders in the template string. In this case, the input variable is text.

The prompt template object can be used to generate a prompt for the LLM. The prompt will be generated by replacing the template string's placeholders with the input variables' values.

The input_variables define what values must be provided each time the template is used. And in this case, it expects a 'text.' With that, this PromptTemplate can be used to summarize any text.

Combine Prompt

Here we are going to create a prompt template that we will use instead of the default combine prompt.

```
combine_custom_prompt='''
Generate a summary of the following text that includes the following elements:

* A title that accurately reflects the content of the text.
* An introduction paragraph that provides an overview of the topic.
* Bullet points that list the key points of the text.
* A conclusion paragraph that summarizes the main points of the text.

Text:`{text}`
'''
```

This prompt is not open-ended, limiting the variety of summaries the LLM can generate. It is very clear, concise, with context, and provides a format for the LLM.

Prompt Template

```
combine_prompt_template = PromptTemplate(
    template=combine_custom_prompt,
```


Again here, the input variable is the text, and the custom prompt will combine the summaries of the first prompt and then summaries it.

Creating the Chain

```
summary_chain = load_summarize_chain (
    llm=llm,
    chain_type='map_reduce',
    map_prompt=map_prompt_template,
    combine_prompt=combine_prompt_template,
    verbose=False
)
summary=summary_chain.run(chunks)
```

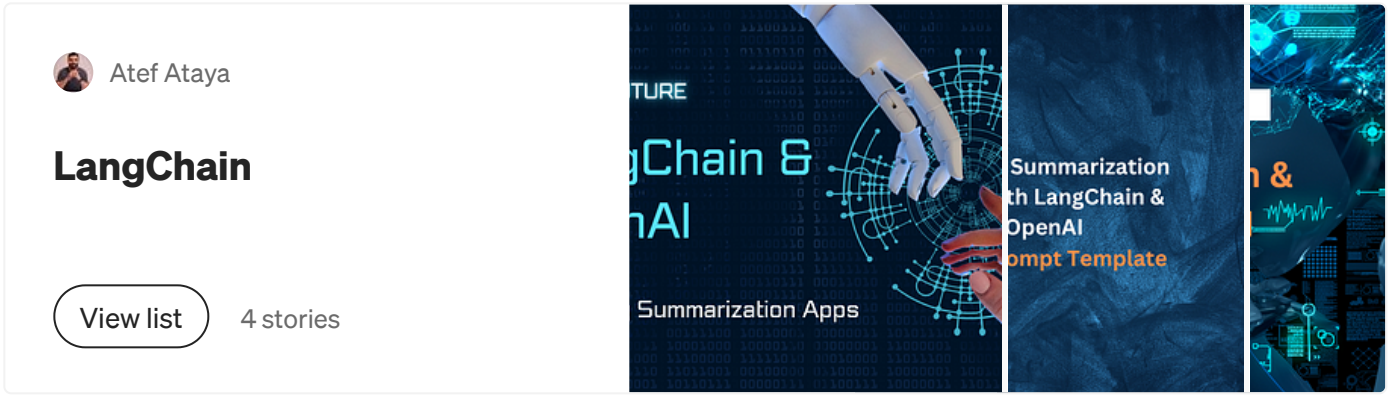
Here the load summarize chain function takes three arguments: the LLM object, the chain type, the map prompt, the combine prompt, and the verbose flag.

The print statement will print the summary of the provided text as follow:

```
Title: Understanding Convolutional Neural Networks (CNNs) for Image Recognition

Introduction:
This text provides an overview of convolutional neural networks (CNNs) and their s

Key Points:
- CNNs are commonly used for image recognition tasks in computer vision.
- They mimic the brain's recognition process by detecting and categorizing feature
- CNNs are crucial in applications like self-driving cars and face recognition.
- They can be used for image classification and object detection.
- CNNs are capable of detecting human emotions in images.
- CNNs scan images by representing black and white images as 2-dimensional arrays
- CNNs exploit knowledge about the specific input type to simplify the network arc

Conclusion:
In conclusion, convolutional neural networks (CNNs) are essential tools for image
```

As we can see, we got an excellent summary using prompt engineering and map reduce. This summary is very clear and concise, and in a readable format.

That's it for this article. If you enjoyed it, please clap and don't forget to follow my account to learn about the latest techniques in the LLMs, like this article. Stay tuned!



Langchain | Prompt Engineering | AI | Artificial Intelligence | Large Language Models

Follow

Written by Atef Ataya

158 Followers

Software dev with a passion for machine learning. Expert in C#,React, .Net Core, & frameworks such as TensorFlow & PyTorch. Sharing insights & tips on Medium.

More from Atef Ataya

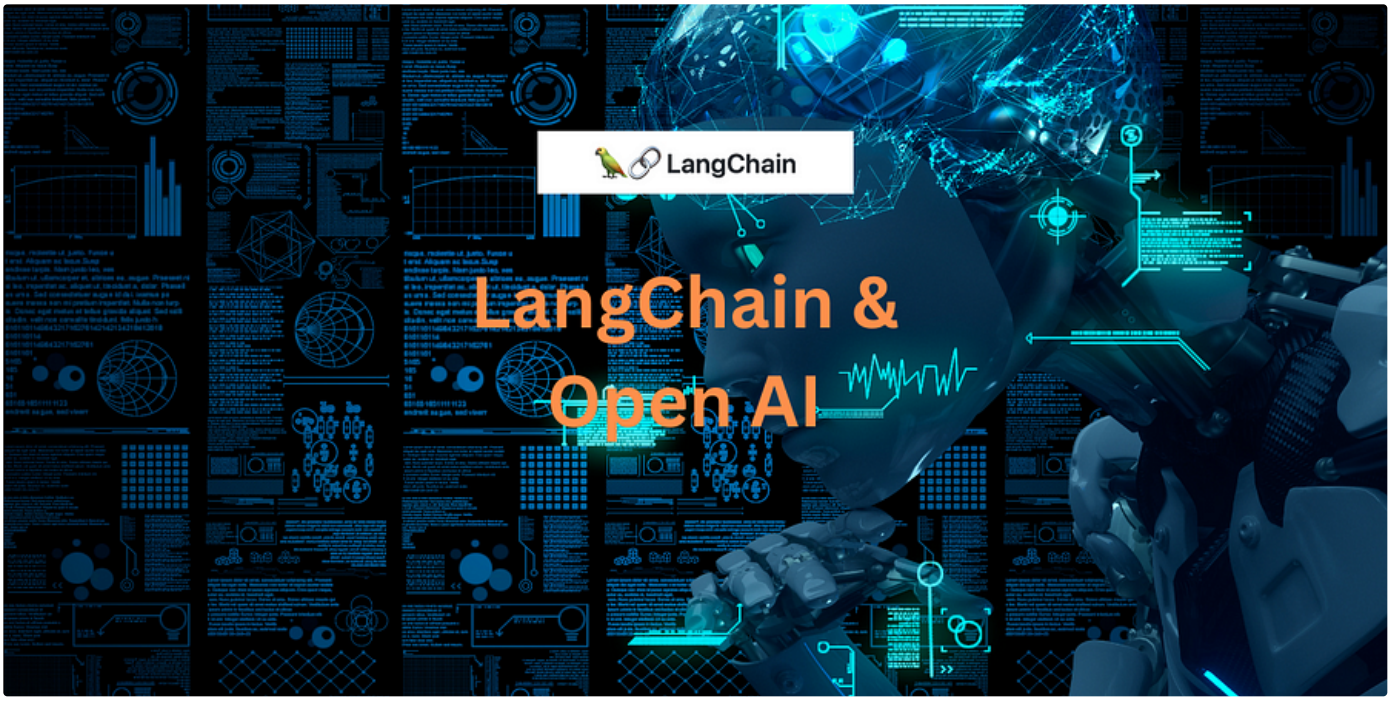
Atef Ataya

Map Reduce: Building Summarization Apps with LangChain and OpenAI

LangChain provides a MapReduce chain that can be used for summarization using a ‘map-reduce’ style workflow. This allows summarizing a…

· 5 min read · Aug 14

114

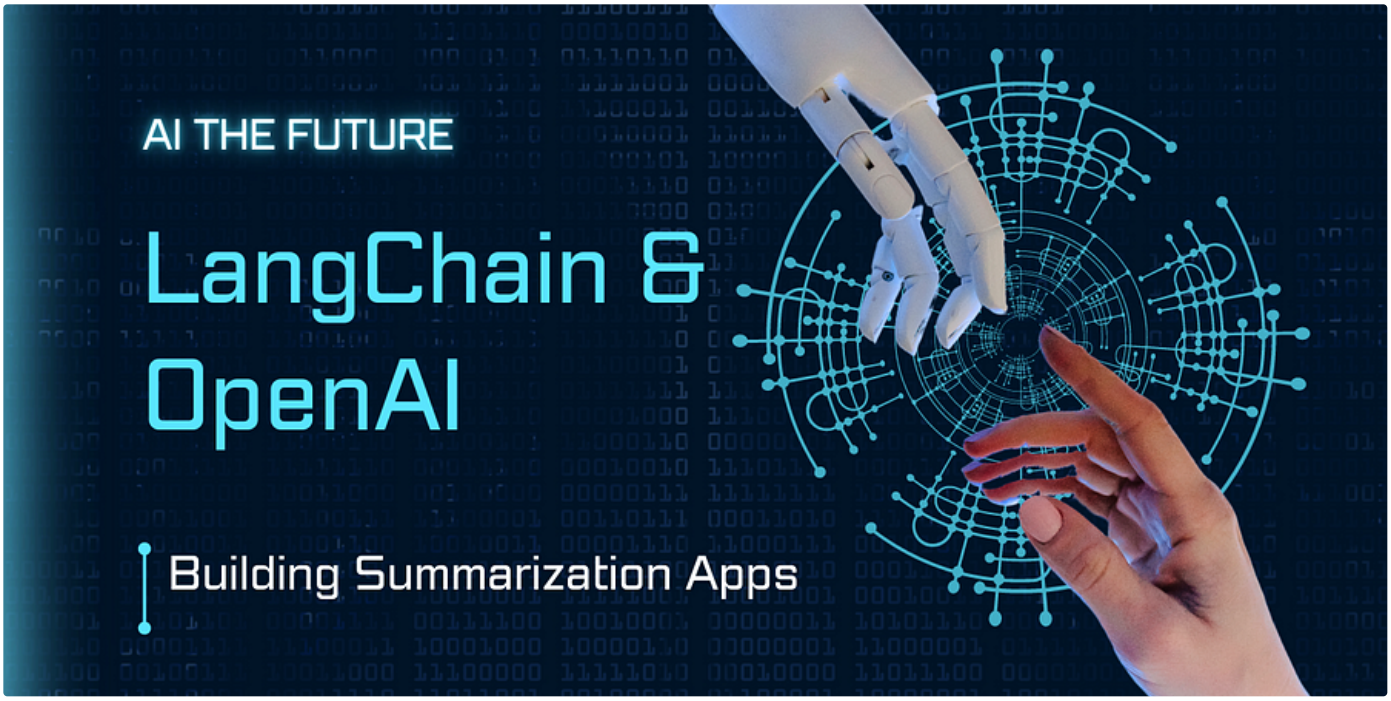


Atef Ataya

Building summarization apps Using StuffDocumentsChain with LangChain & OpenAI

In this story, we will build a summarization app using Stuff Documents Chain. But first let us talk about what is Stuff Documents Chain.

· 4 min read · Aug 11



Atef Ataya

Building Summarization Application with LangChain and OpenAI: Using Basic Prompt

In this article we are going to build a basic summarization application using LangChain basic prompt. Whether you are a beginner or...

· 7 min read · Aug 7

Atef Ataya

Building Summarization Apps with LangChain and OpenAI using Prompt Template

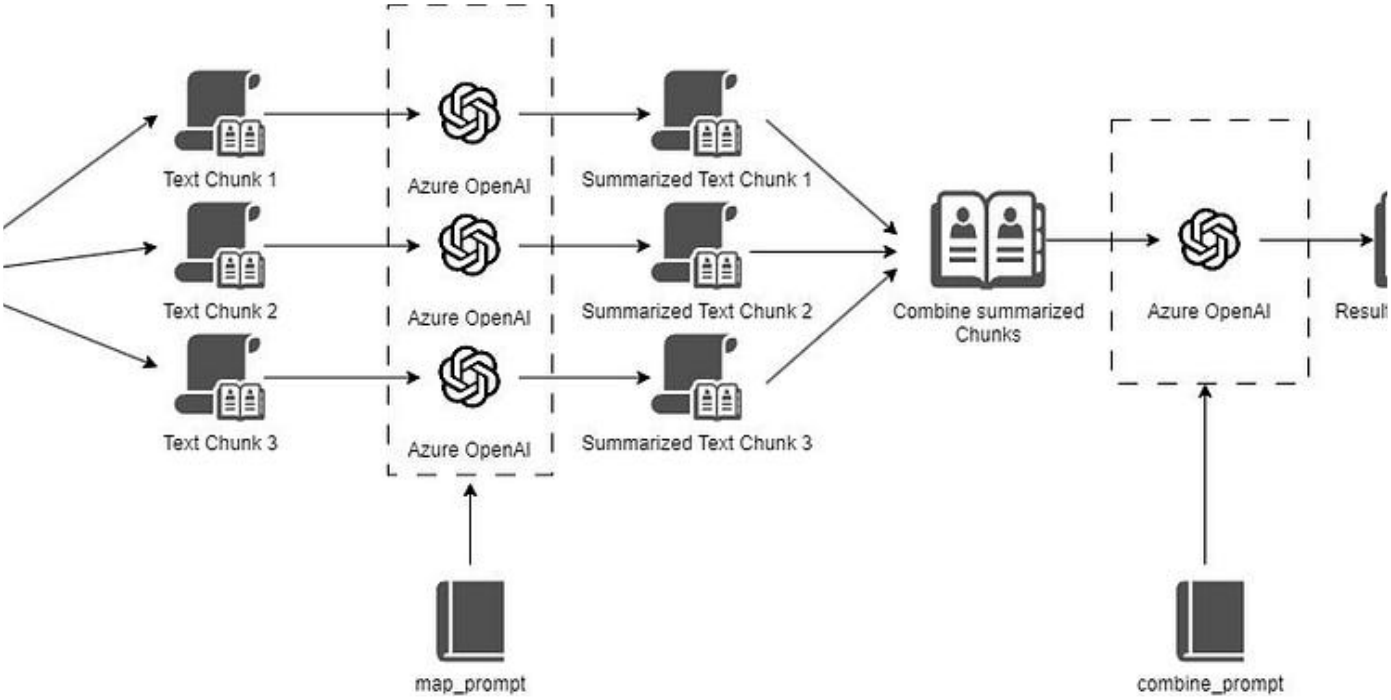
In this article I will show you how to generate a summary of text or paragraph and then translate it into another language. We will use the...

· 6 min read · Aug 8

151

See all from Atef Ataya

Recommended from Medium



Bluetick Consultants

Advanced Techniques in Text Summarization: Leveraging Generative AI and Prompt Engineering

Generative AI and Language Models like OpenAI's GPT-3.5 have emerged as powerful tools for text summarization. In this blog, we will explore

9 min read · Jun 3

6 1

Mark Chen in Towards Data Science

How to Validate OpenAI GPT Model Performance with Text Summarization

Part 1 of a study on generative AI usage and testing

9 min read · Apr 5

300 5

Lists

AI Regulation

6 stories · 127 saves

Generative AI Recommended Reading

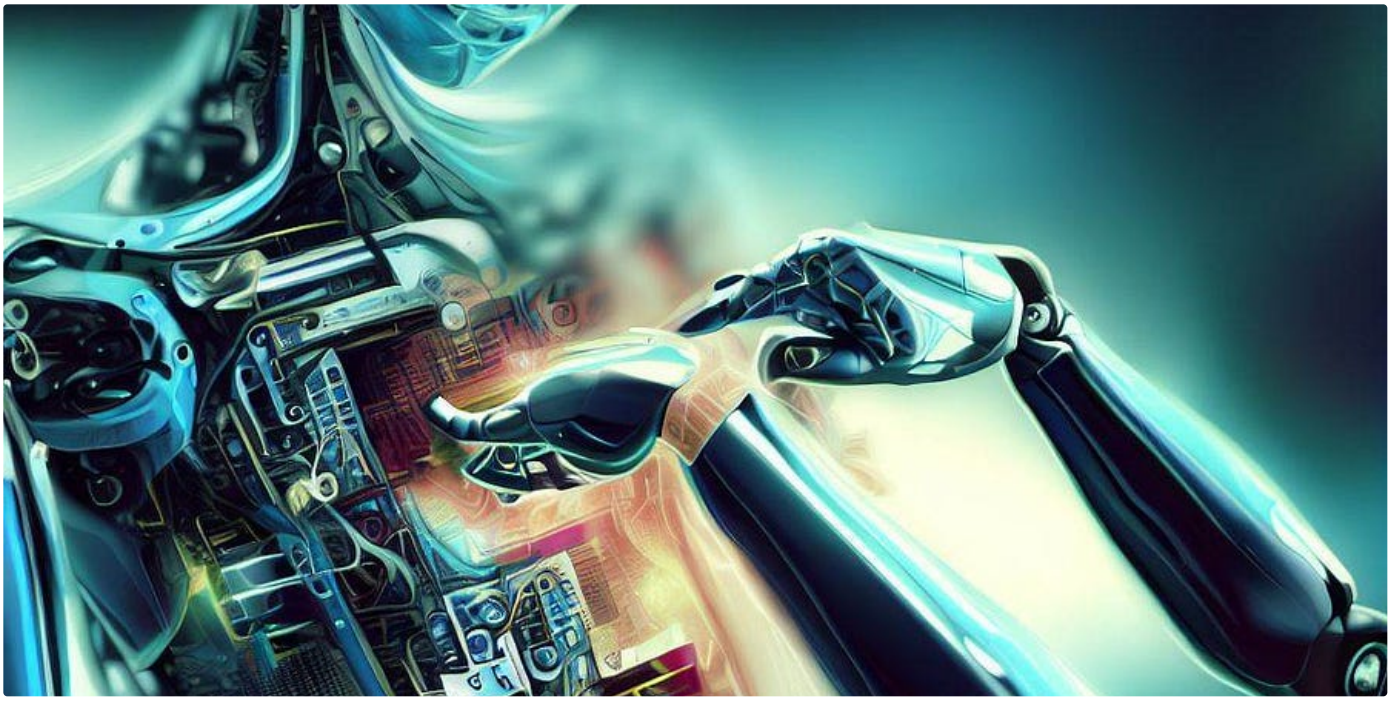
52 stories · 244 saves

Natural Language Processing

634 stories · 242 saves

What is ChatGPT?

9 stories · 176 saves



Charles Suárez in AI Mind

Building a Custom Chat Agent for Document QA with Langchain, GPT-3.5 and Pinecone

We'll learn how to build a custom chat agent using Langchain, add memory, create a custom prompt template, its output parser and a QA tool.

16 min read · Sep 3

127

Manmeet Kaur

LangChain

An interesting LLM (Large Language Model) framework, which was launched recently in October 2022, is Lang Chain by a machine learning...

1 min read · Sep 12



Thu Ya Kyaw in Google Cloud - Community

How to Use LLMs to Generate Concise Summaries

Large language models (LLMs) are a type of artificial intelligence (AI) that can be used to generate text. They are trained on massive...

4 min read · May 29

27 2

Yvann in Better Programming

Build a Chatbot on Your CSV Data With LangChain and OpenAI

Chat with your CSV file with a memory chatbot 🤖 — Made with Langchain 🦜 and OpenAI 🧠

5 min read · Jun 2

1K 24