Photo by Aleks Dorohovich on Unsplash

Member-only story

Open in app

Search Medium

Lucas Soares · Follow

Published in Geek Culture

6 min read · Apr 27, 2021

Listen Share More

When access to the GPT-3 API was released to researchers and engineers (upon request) I immediately requested it so I could see what kind of interesting tools one could write and what kind of interesting research questions could be asked.

Upon trying the API and the awesome tools that come with it, I realized that one of my favorite applications of GPT-3 was for ***paper summarization,*** so I decided to test it out

Today I will show you how to build a simple python tool to summarize papers directly from their arxiv address.

Photo by Seven Shooter on Unsplash

Steps to summarize a paper with GPT-3

The process itself is quite simple:

1. Download the paper

2. Convert from pdf to text

3. Feed the text to the GPT-3 model using the openai api

4. Show the summary

1. Download the paper

First let's import our dependencies

```
import openai
import wget
import pathlib
import pdfplumber
import numpy as np
```

[Updated 2021 — the GPT-3 model is now available for all]

Here you will have to install `openai` for interfacing with GPT-3 in case you have an API key, if you don't have it you can get started here.

You will also need `wget` for downloading the pdf from the arxiv page and `pdfplumber` for converting the pdf to text:

```
pip install openai
pip install wget
pip install pdfplumber
```

Now, let's write a function that downloads a pdf from an arxiv address, the paper I will be using is *'Understanding training and generalization in deep learning by Fourier analysis'* by Zhi-Qin John Xu.

```
def getPaper(paper_url, filename="random_paper.pdf"):
    """
    Downloads a paper from it's arxiv page and returns
    the local path to that file.
    """
    downloadedPaper = wget.download(paper_url, filename)
```

```
    downloadedPaperFilePath = pathlib.Path(downloadedPaper)

    return downloadedPaperFilePath
```

Here, I am using the `wget` package to directly download the pdf and return a path to the downloaded file.

2. Convert from pdf to text

Now, I will write another function to convert the pdf to text so that GPT-3 can actually read it.

```
paperFilePath = "random_paper.pdf"
paperContent = pdfplumber.open(paperFilePath).pages

def displayPaperContent(paperContent, page_start=0, page_end=5):
    for page in paperContent[page_start:page_end]:
        print(page.extract_text())
displayPaperContent(paperContent)

# Output
Understanding training and generalization in deeplearning by Fourier
analysisZhi-
QinJohnXu*8NewYorkUniversityAbuDhabi1AbuDhabi129188,UnitedArabEmirates
0zhiqinxu@nyu.edu2 voN Abstract 92 Background: It is still an open
research area to theoretically understand why
DeepNeuralNetworks(DNNs)—equippedwithmanymoreparametersthantrain- ]
ing data and trained by (stochastic) gradient-based methods—often
achieve re-Gmarkably low generalization error. Contribution: We study
DNN training byL Fourier analysis. Our theoretical frameworkexplains:
i) DNN with (stochastic). gradient-based methods often endows low-
frequency components of the targetsc function with a higher priority
during the training; ii) Small initialization leads[ to good
generalizationability of DNN while preservingthe DNN’s ability to fit
any function. These results are further confirmed by experiments of
DNNs
...
...
```

Here, I extracted the text per page from the paper and wrote a function `displayPaperContent` to show the corresponding content. There are issues with the conversion that I won’t address in this article because I want to focus just on the summarization pipeline.

3. Feed the text to the GPT-3 model using the openai api

Now, for the fun stuff, let's write a function that get's the paper content and feeds it to the GPT-3 model using openai's api:

```
def showPaperSummary(paperContent):
    tldr_tag = "\n tl;dr:"
    openai.organization = 'organization key'
    openai.api_key = "your api key"
    engine_list = openai.Engine.list() # calling the engines available
from the openai api

    for page in paperContent:
        text = page.extract_text() + tldr_tag
        response =
openai.Completion.create(engine="davinci",prompt=text,temperature=0.3,
            max_tokens=140,
            top_p=1,
            frequency_penalty=0,
            presence_penalty=0,
            stop=["\n"]
        )
        print(response["choices"][0]["text"])
```

Let's unpack what is happening here:

```
tldr_tag = "\n tl;dr:"
```

In this step I am writing the tag so that the GPT-3 model knows when the text stops and when it should start the completion (which in this case is a summary).

```
openai.organization = "the openai organization key"
openai.api_key = "your api key"
engine_list = openai.Engine.list() # calling the engines available  #
from the openai api
```

Here, I am setting up the environment for using the openai API.

```
for page in paperContent:
        text = page.extract_text() + tldr_tag
```

Now, I am extracting the text from each page and feeding it to the GPT-3 model. In the future I want to write an extension to this script so that it can get paragraph by paragraph, so that the model could see semantically connected chunks from the text.

```
response =
openai.Completion.create(engine="davinci",prompt=text,temperature=0.
            max_tokens=140,
            top_p=1,
            frequency_penalty=0,
            presence_penalty=0,
            stop=["\n"]
        )
        print(response["choices"][0]["text"])
```

Finally, I am feeding the text to the model setting a max tokens of 140 for the response (half the size of a tweet) for each page and printing that to the terminal.

4. Show the summary

Ok great! Now that we have our model set up, let's run it and see the results:

```
paperContent = pdfplumber.open(paperFilePath).pages
showPaperSummary(paperContent)

# Output:

The power spectrum of the tanh function exponentially decays w.r.t.
frequency.

Thegradient-basedtrainingwiththeDNNwithonehiddenlayerwithtanhfunction
Thetotallossfunctionofasinglehiddenlayer DNNis


 The initial weights of a deep neural network are more important than
the initial biases.
 We can use Fourier analysis to understand the gradient-based
optimization of DNNs on real data and pure noise. We found that DNNs
have a better generalization ability when the loss function is flatter.
We also found that the DNN generalization ability is better when the
DNN parameters are smaller.
```

```
 The code is available at https://github.com/yuexin-wang/DeepLearning-
Music .

 Theorem 3 is true.
 We can use the same method to prove the theorem.
 The DNN is a very powerful tool that can be used for many things. It
is not a panacea, but it is a very powerful tool.
 Deep learning is only one of many tools for building intelligent
systems.
```

And there it is! This is so cool! Due to issues with the pdf conversion, two page summaries appeared glued together but overall it gives an ok description of the pa page by page, even highlighting the source code link!

Putting it all together

The full code is this:

```
import openai
import wget
import pathlib
import pdfplumber
import numpy as np

def getPaper(paper_url, filename="random_paper.pdf"):
    """
    Downloads a paper from it's arxiv page and returns
    the local path to that file.
    """
    downloadedPaper = wget.download(paper_url, filename)
    downloadedPaperFilePath = pathlib.Path(downloadedPaper)

    return downloadedPaperFilePath


def showPaperSummary(paperContent):
    tldr_tag = "\n tl;dr:"
    openai.organization = 'API KEY org'
    openai.api_key = "your openAI key"
    engine_list = openai.Engine.list()

    for page in paperContent:
        text = page.extract_text() + tldr_tag
        response =
openai.Completion.create(engine="davinci",prompt=text,temperature=0.3,
            max_tokens=140,
```

```
            top_p=1,
            frequency_penalty=0,
            presence_penalty=0,
            stop=["\n"]
        )
        print(response["choices"][0]["text"])


paperContent = pdfplumber.open(paperFilePath).pages
showPaperSummary(paperContent)
```

Thoughts on Summarization

I think summarization models are great. They will never replace the important process of actually reading an entire paper, but they can serve as a ***tool to explore a wider range of interesting scientific discoveries.***

This post was more a proof of concept, mostly because the issues with formatting the text converted from the pdf and the actual quality of the summarization are still things to tweak quite a bit. However, I feel like we have come a long way and now, more than ever, we are getting closer to having something that will actually allow us to process a bigger array of scientific information.

[Update 22/11/2021]

I made a youtube video about this that you can check out here:

If you want to dive deeper into natural language processing and text summarization, check out this course:

> *This is an affiliate link, if you use the course I get a small commission, cheers! :)*

- **Natural Language Processing for Text Summarization**

If you liked this post connect with me on Twitter, LinkedIn and follow me on Medium. Thanks and see you next time! :)

Data Science | Artificial Intelligence | Python | Programming | Software Development

Follow

Written by Lucas Soares

1.7K Followers · Writer for Geek Culture

Machine Learning Engineer. I write about AI | Data Science | Productivity & Learning. Join Medium at: https://lucas-soares.medium.com/membership

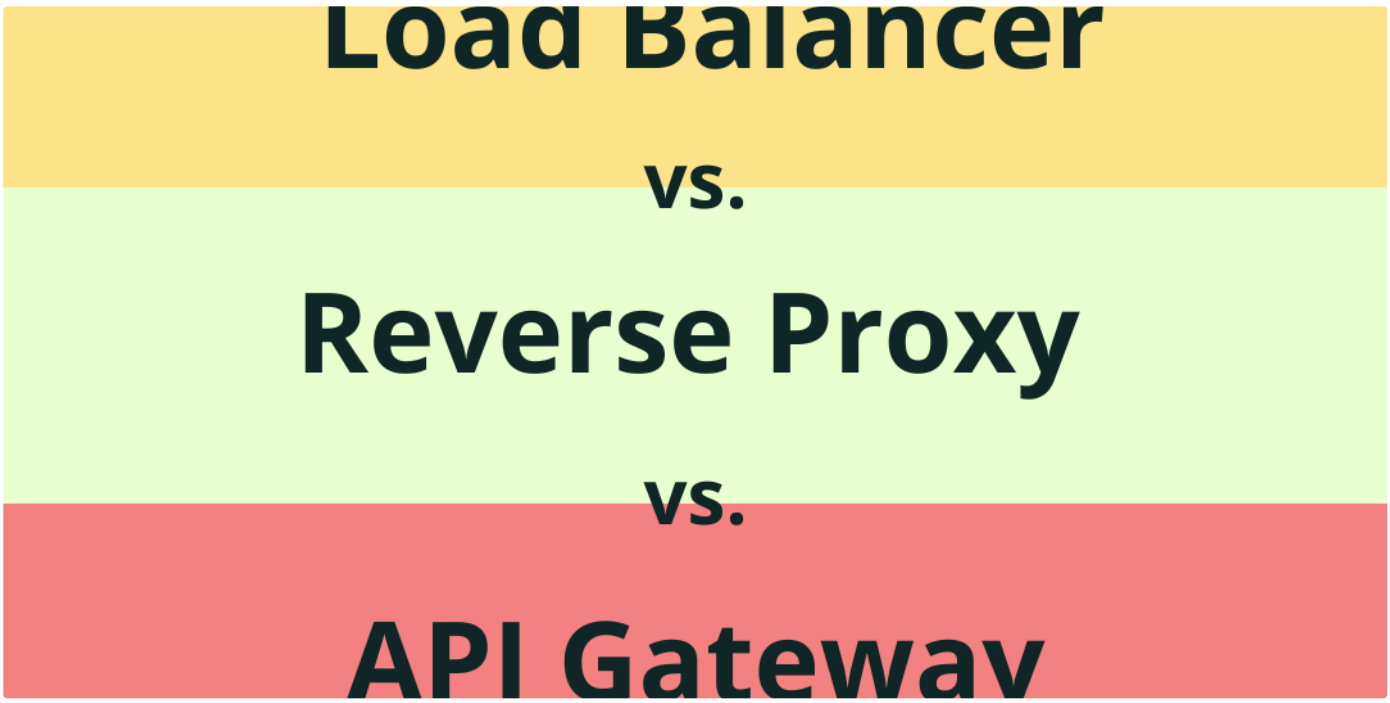
More from Lucas Soares and Geek Culture

Lucas Soares in MLearning.ai

7 Ways to Use the ChatGPT Code Interpreter Plugin

My favorite ways to use the ChatGPT Code Interpreter

· 4 min read · Jul 18



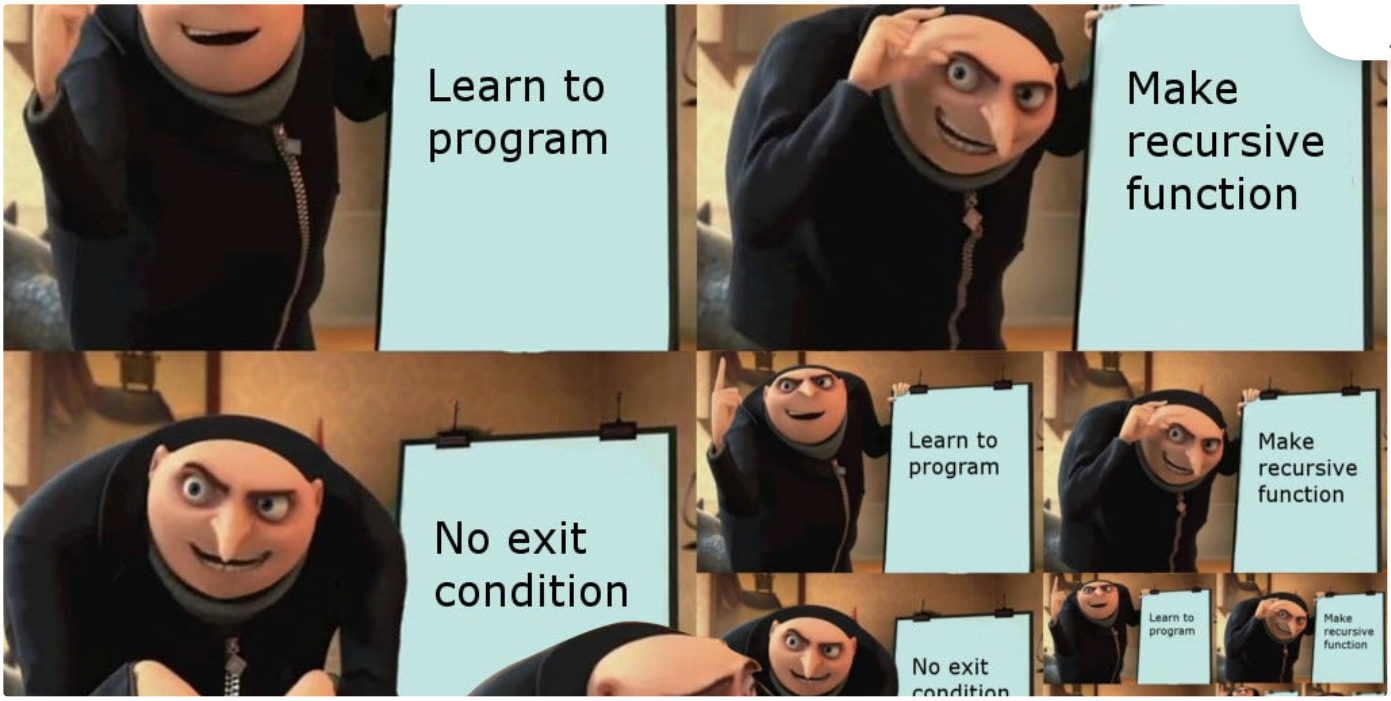
Arslan Ahmad in Geek Culture

Load Balancer vs. Reverse Proxy vs. API Gateway

Understanding the Key Components for Efficient, Secure, and Scalable Web Applications.

12 min read · May 17

1.4K 9



Joanna in Geek Culture

Understanding the 10 Most Difficult Python Concepts

Python has a simple syntax and is easy to learn, making it an ideal choice for beginners. However, as you delve deeper into Python, you may...

· 8 min read · Apr 3

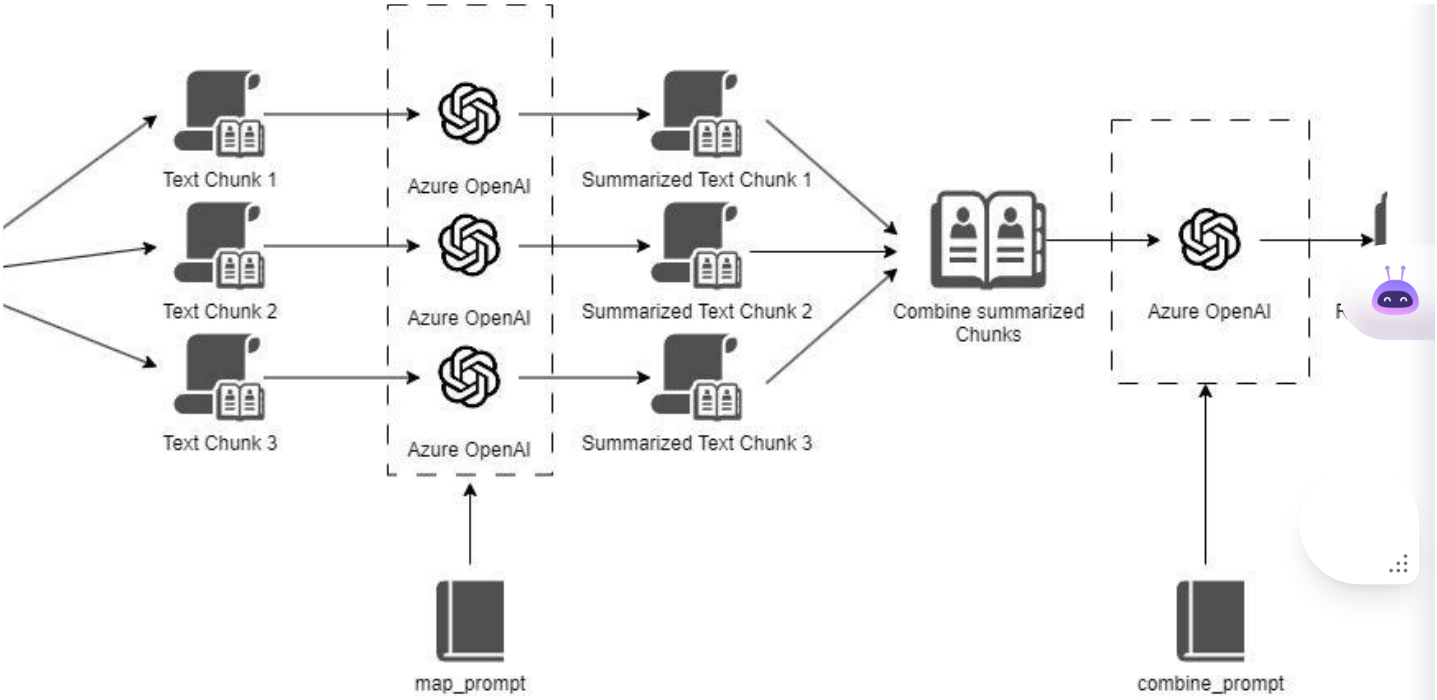
656 3

Lucas Soares in MLearning.ai

Summarizing and Querying Multiple Research Papers with LangChain

An easy and simple way to boost research productivity using LangChain

· 3 min read · May 7



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

Tushit Dave

Llama2 and Text Summarization

Unlocking the Power of Llama2 for Local Multi-Document Summarization

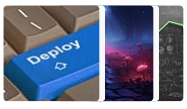
8 min read · Sep 8

37 1

Lists

Coding & Development

11 stories · 189 saves



Predictive Modeling w/ Python

20 stories · 414 saves

General Coding Knowledge

20 stories · 359 saves

ChatGPT

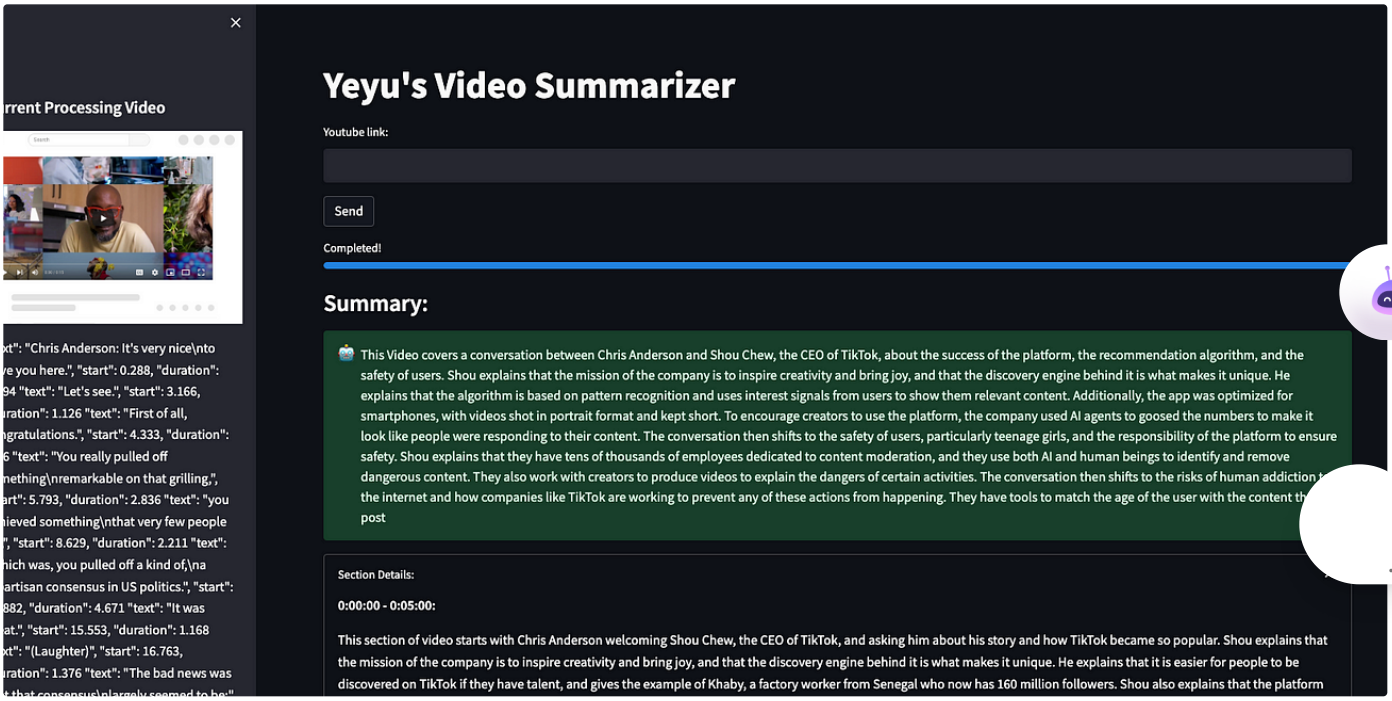
21 stories · 161 saves

Maximilian Vogel in MLearning.ai

The ChatGPT list of lists: A collection of 3000+ prompts, examples, use-cases, tools, APIs...

Updated Sep-09, 2023. Added new prompt engineering courses, masterclasses and tutorials.

10 min read · Feb 7



Yeyu Huang in Level Up Coding

How To Create A Video Summarizer Powered By AI, In 20 Minutes

A quick guide for building a website that generates summaries for any Youtube video

· 11 min read · May 7

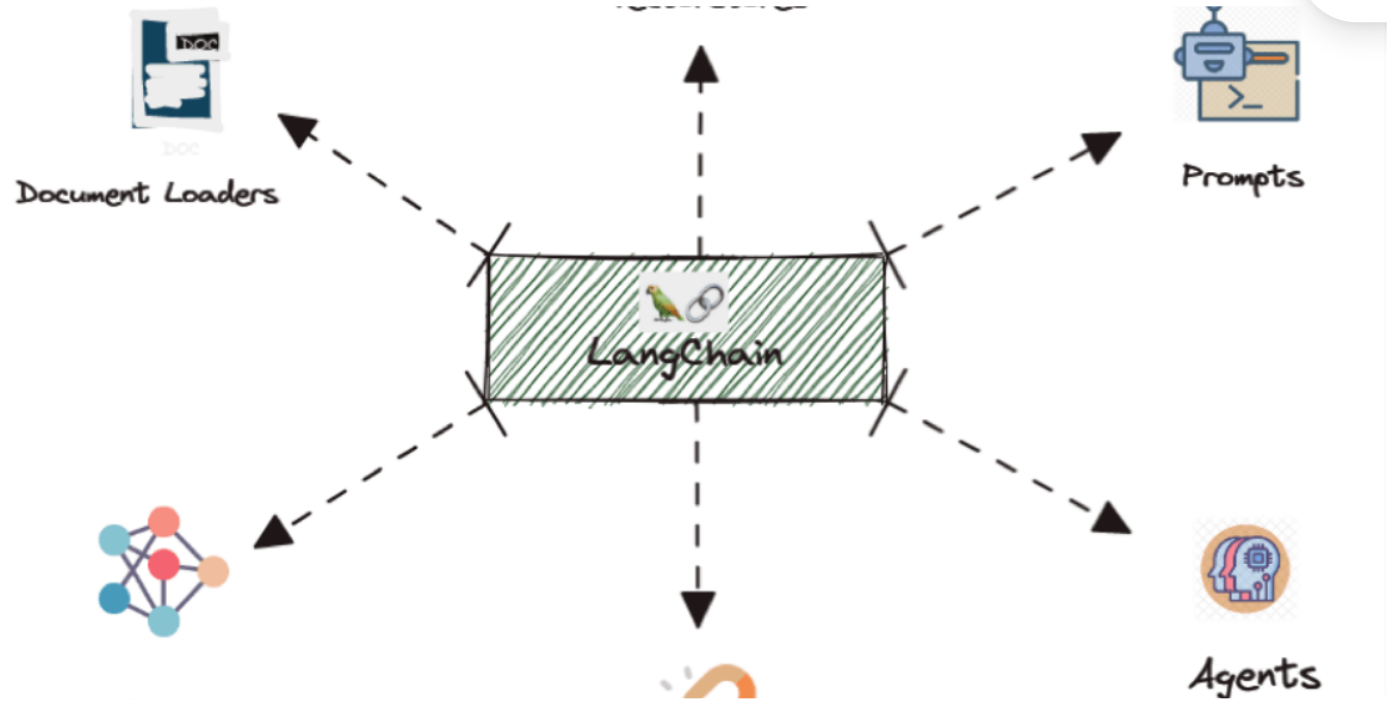
Thu Ya Kyaw in Google Cloud - Community

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



Zeeshan Malik

Connecting ChatGPT with your own Data using Llama Index and LangChain

In the last three months, there has been a rapid increase in the use of Large Language Models (LLMs) for a variety of applications, such as...

5 min read · Jun 11

88 2