通过 OpenAI 和 Langchain 构建 Arxiv 论文摘要 Twitter 机器人

arXiv 是一个收集物理学、数学、计算机科学、生物学与数理经济学的论文预印本的网站，成立于1991年8月14日。截至2008年10月，arXiv 已收集超过50万篇预印本；至2014年底，藏量达到1百万篇。截至2016年10月，每月提交量超过10,000篇。

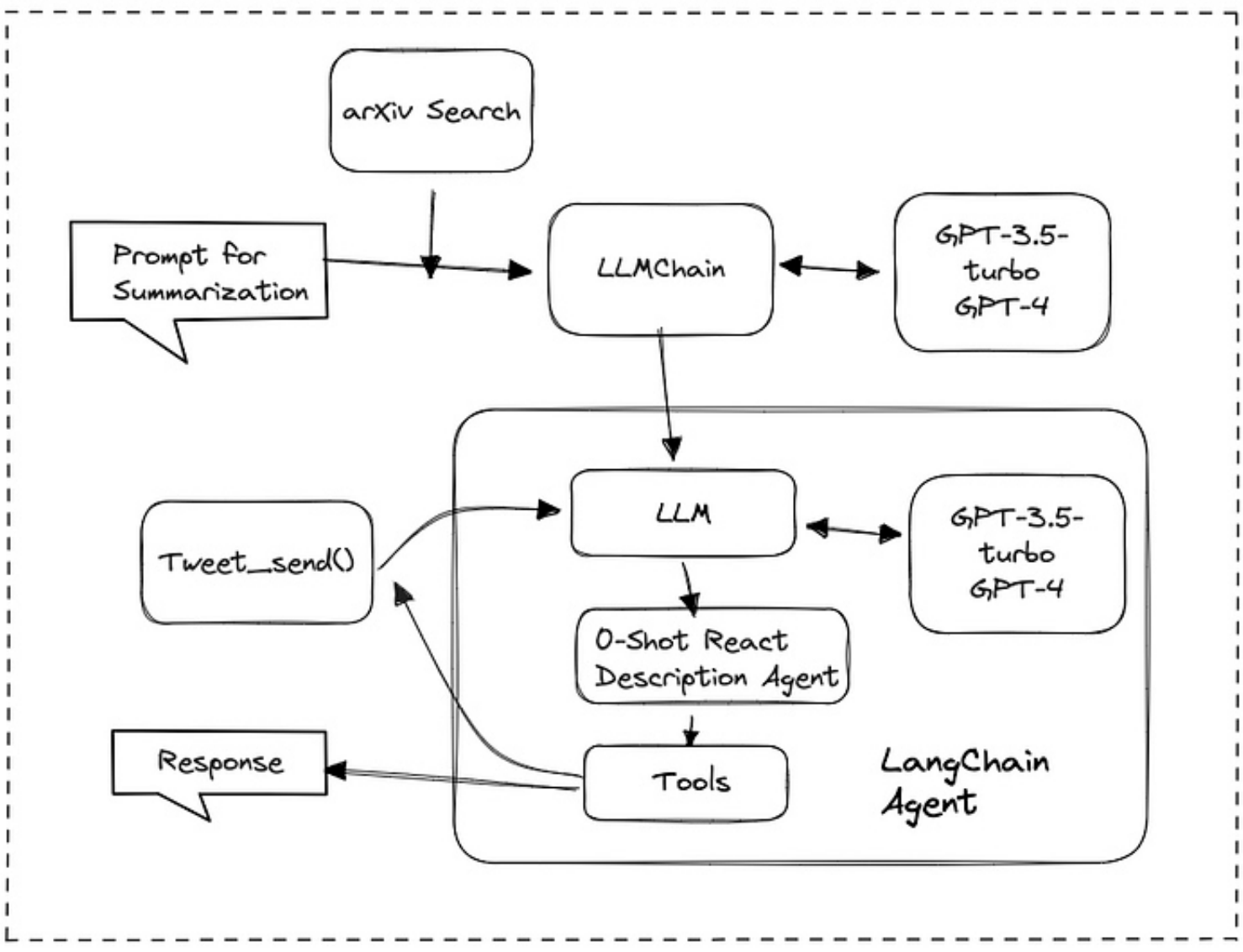
arXiv 网站的研究论文每天都在更新，跟上 arXiv 最新科学成就的节奏就成了一件具有挑战的事情。生在这个年代的我们是幸运的，跟上 AI 的潮流，通过使用大型语言模型及其支持应用程序框架（如 Langchain），可以实现自动跟踪 arXiv 论文的新版本。在本教程中，将介绍如何设计一个 Twitter 机器人，发布新论文摘要的推文，让研究人员和人工智能爱好者快速了解最有趣的科学成就或发现。

1、总体设计

先感受一下机器人发送的演示推文：

Twitter 机器人在功能上称为 `arXiv 论文摘要器`，由几个无缝协作的关键组件组成。这些组件包括用于检索新论文发布的 `arXiv API`、用于摘要的 `LLMChain` 以及带有自定义推文工具的 `Langchain` 代理，用于执行整个发布过程。

为了更好地理解该机器人内的信息流和操作，下面是整体的架构图：



在此演示应用程序中，将语言模型在不同步骤中运行两次。第一次运行是使用普通链根据论文摘要生成摘要，第二次运行是要求代理执行推文帖子。显然，第二次运行可以用没有任何 AI 参与的纯程序代码代替，但是，为了演示如何最好地利用生成式 AI 来完成未来更复杂的任务，值得将此任务包装到 Langchain 代理中。

2、代码模块

现在，就来逐步探索使这个推文机器人的代码工作流程。将深入研究每个步骤，提供代码片段和解释，以确保顺利实施。

arXiv API

arXiv 是康奈尔大学图书馆的一个流行的在线存储库，存储物理、数学、计算机科学等各个领域的科学论文。它为研究人员在正式同行评审和在传统科学期刊上发表之前分享他们的工作提供了一个平台。arXiv 上托管的论文可以免费向公众开放，让研究人员和技术爱好者能够及时了解最新的研究进展。

arXiv 还提供对其资源的 API 访问，可以通过 Python 调用API定期下载所需的论文及其关键数据。

安装软件包：

```
pip install arxiv
```

搜索关于 `language model` 的最新论文。

```
import arxiv
def arxiv_search():
    summary = ''
    search = arxiv.Search(
      query = "language model",
      max_results = 1,
      sort_by = arxiv.SortCriterion.SubmittedDate
    )
    for result in search.results():
       summary = f"Title: {result.title}, Abstract: {result.summary}, URL: {result.entry_id}"

    return summary
```

代码通过 API 在 arXiv 数据库上执行 `search()` 检索与查询匹配关于 `language model` 的最新论文，结果按提交日期排序。

然后迭代搜索结果，对于每个结果，它将论文的标题、摘要和 URL 附加到将用于后续文本摘要步骤的 `summary` 字符串。

> 有关 arXiv API 的更详细用法，可以在这里找到它们。

使用 LLMChain 进行摘要

现在，将实现一个简单的 LLM 链来自动处理动态提示，要求 `gpt-3.5-turbo` 模型在 `arXiv` 抽象上下文上生成摘要。

安装软件包：

```
pip install langchain openai
```

将 OpenAI API 密钥设置为环境。

```
os.environ["OPENAI_API_KEY"] = "{Your_API_Key}"
```

创建一个好的提示模板（始终是最重要的）。

```
prompt_tweet_template = """
You are a AI technology blogger who write regular tweets for shortly summarizing new papers on Arxiv about language
model topics. You write summary for tweets based on text with content of Title, Abstract, URL: ##start {text} ##end
Your write a 50-word quick summary including key information from the Title and Abstract, then, attach the URL after summary.
Summary:
"""
```

使用模型 `gpt-3.5-turbo` 和提示模板 `PROMPT_TWEET` 创建 `LLMChain`。

```
from langchain import PromptTemplate, LLMChain
from langchain.chat_models import ChatOpenAI

PROMPT_TWEET = PromptTemplate(template=prompt_tweet_template, input_variables=["text"])
llm = ChatOpenAI(temperature=0.5)
tweet_chain = LLMChain(llm=llm, prompt=PROMPT_TWEET,verbose=True)
```

从 `arxiv_search()` 提供最新 arXiv 论文的摘要字符串。

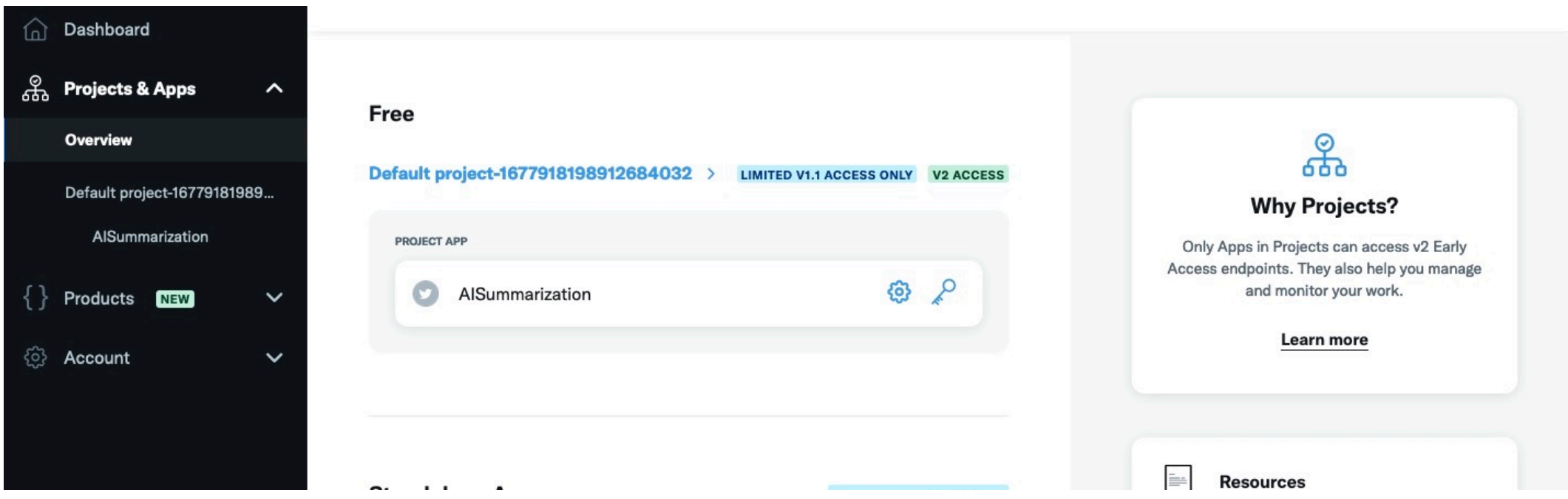
```
new_paper = arxiv_search()
tweet_body = tweet_chain.run(new_paper)
```

现在有了一条不错的推文，其中包含 `arXiv` 论文摘要 `tweet_body` ，接下来就准备发送。

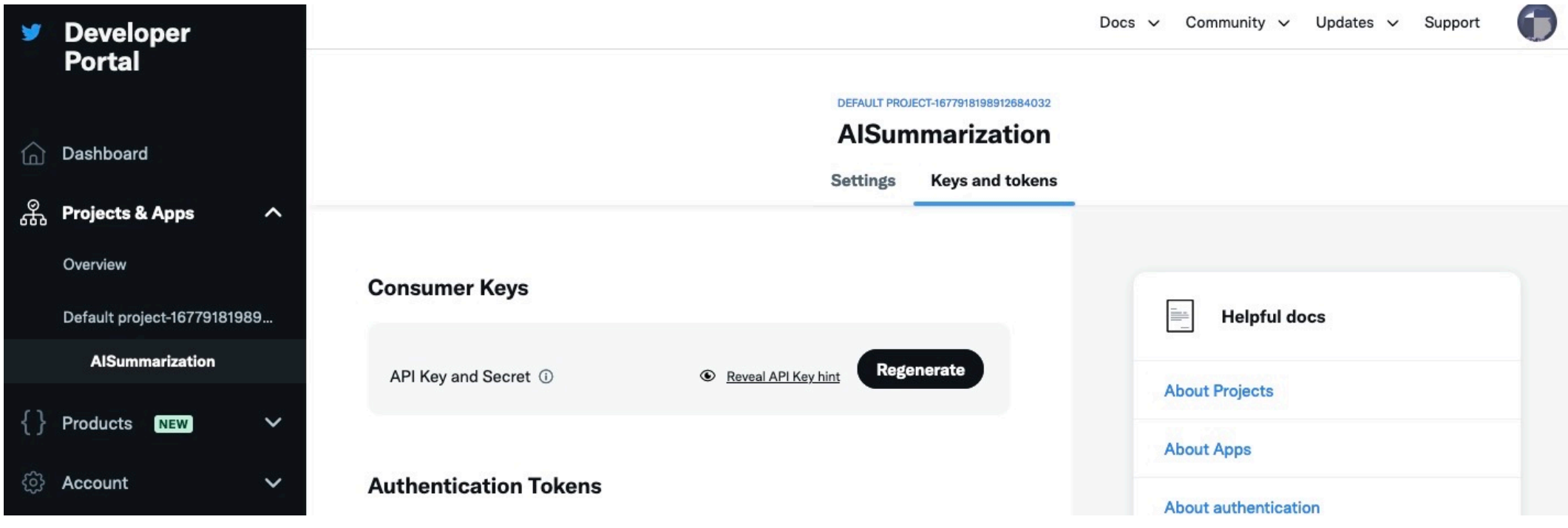
Twitter API

要获得 Twitter API 的许可，必须登录 Twitter 开发者平台创建一个项目及其应用程序，以方便机器人稍后调用它们的 API。

开发者平台的 URL：https://developer.twitter.com/en/portal/。



如果开发者免费账户，则只能创建包含一个应用程序的项目。重命名项目应用程序后，找到`钥匙`图标，然后单击，将重定向到该应用程序的 `Keys and tokens` 页面。



单击 `Generate` 或 `Regenerate` 按钮生成密钥，然后将 `consumer_key` 和 `consumer_secret` 复制到将在后续步骤中使用的安全位置。

以下代码演示了如何通过密钥和 `secret` 授权获得对 Twitter API 的完全访问权限。

安装软件包

```
pip install requests requests_oauthlib
```

通过授权来初始化 Twitter 会话。

```
def tweet_init():
    consumer_key = "{Your_consumer_key}"
    consumer_secret = "{Your_consumer_secret}"

    # Get request token
    request_token_url = "https://api.twitter.com/oauth/request_token?oauth_callback=oob&x_auth_access_type=write"
    oauth = OAuth1Session(consumer_key, client_secret=consumer_secret)
    print(oauth)
    try:
        fetch_response = oauth.fetch_request_token(request_token_url)
    except ValueError:
```

```
        print(
            "There may have been an issue with the consumer_key or consumer_secret you entered."
        )

    resource_owner_key = fetch_response.get("oauth_token")
    resource_owner_secret = fetch_response.get("oauth_token_secret")
    print("Got OAuth token: %s" % resource_owner_key)

    # Get authorization
    base_authorization_url = "https://api.twitter.com/oauth/authorize"
    authorization_url = oauth.authorization_url(base_authorization_url)
    print("Please go here and authorize: %s" % authorization_url)
    verifier = input("Paste the PIN here: ")

    # Get the access token
    access_token_url = "https://api.twitter.com/oauth/access_token"
    oauth = OAuth1Session(
        consumer_key,
        client_secret=consumer_secret,
        resource_owner_key=resource_owner_key,
        resource_owner_secret=resource_owner_secret,
        verifier=verifier,
    )
    oauth_tokens = oauth.fetch_access_token(access_token_url)

    access_token = oauth_tokens["oauth_token"]
    access_token_secret = oauth_tokens["oauth_token_secret"]

    oauth = OAuth1Session(
        consumer_key,
        client_secret=consumer_secret,
        resource_owner_key=access_token,
        resource_owner_secret=access_token_secret,
    )
    return oauth

mytweet = tweet_init()
```

此代码片段设置使用 `OAuth 1.0` 访问 Twitter API 的身份验证过程。它涉及获取授权所需令牌的几个步骤。

- 首先使用 `fetch_request_token()` 方法从 Twitter 的 API 请求临时请求令牌 `resources_owner` `token` 和 `secret`。
- 生成一个临时 URL，用于提供 PIN 码，希望用户在终端提示中输入如下所示的内容：

> 请前往此处授权：`https://api.twitter.com/oauth/authorize?oauth_token=jvPdHgAAAAABofwiAAABiRDFU_M` 将 `PIN` 码粘贴到此处：`{Input the PIN display on above link}`

- 使用 PIN 码作为 `verifier`，与其他两个 `token/secret` 对一起请求一对访问 `token/secret`。
- 最后，使用获得的 `access token` 和 `secret` 重新配置 `OAuth1Session` 实例。此重新配置允许使用 Twitter 帐户对 Twitter API 的后续请求进行身份验证。

在应用程序被授予对 API 的访问权限后，只需编写一个函数 `tweet_send()` 即可通过授权实例 `mytweet` 发布包含有效负载内容的推文。

```
def tweet_send(payload):

    payload_json = {"text": payload}
    # Making the request
    response = mytweet.post(
        "https://api.twitter.com/2/tweets",
        json=payload_json,
    )

    if response.status_code != 201:
        raise Exception(
            "Request returned an error: {} {}".format(response.status_code, response.text)
        )

    return  response.status_code
```

Langchain 代理创建

最后一步是创建一个运行 ReAct 来发布推文的 Langchain 代理。

导入必要的模块。

```
from langchain.agents import AgentType, initialize_agent
from langchain.tools import BaseTool, StructuredTool, Tool, tool
from pydantic import BaseModel, Field
from typing import Optional, Type

from langchain.callbacks.manager import (
    AsyncCallbackManagerForToolRun,
    CallbackManagerForToolRun,
)
```

创建从 BaseTool 派生的名为 `TweetSendTool` 的工具来调用外部函数 `tweet_send(payload)`。

```
class TweetSendInput(BaseModel):
    payload: str = Field(..., description="This is the content of tweet to be sent.")

class TweetSendTool(BaseTool):
    name = "Tweet Send"
    description = "useful for when you need to send a tweet"

    def _run(
        self, payload: str, run_manager: Optional[CallbackManagerForToolRun] = None
    ) -> str:
        """Use the tool."""
        return tweet_send(payload)

    async def _arun(
        self, payload: str, run_manager: Optional[AsyncCallbackManagerForToolRun] = None
    ) -> str:
        """Use the tool asynchronously."""
        raise NotImplementedError("Tweet send does not support async")

    args_schema: Type[BaseModel] = TweetSendInput
```

从在前面的步骤中创建的现有工具和 llm 创建代理。

```
tools = [TweetSendTool()]
agent = initialize_agent(
    tools, llm, agent=AgentType.ZERO_SHOT_REACT_DESCRIPTION, verbose=True
)
```

构建一个像样的提示模板，用于从备用 arXiv 摘要 `tweet_body` 到推文发布方法的连接。

```
instructions = f"""
    Post a tweet with the summary of Arxiv paper, use the common Twitter expression style, and with '#AI #arXiv', and provide URL at the end.
    The summary: {tweet_body}
"""
```

运行代理。

```
agent.run(instructions)
```

每次代理运行时，Twitter 上都会发布一条推文，其中总结了最新 arXiv 论文中有关语言模型主题的关键信息。不足的是每次都需要授权 PIN 码，免费的 Twitter API 没办法实现自动执行发布。