Emmanuel Onwuegbusi
Posted on 19 oct. 2023

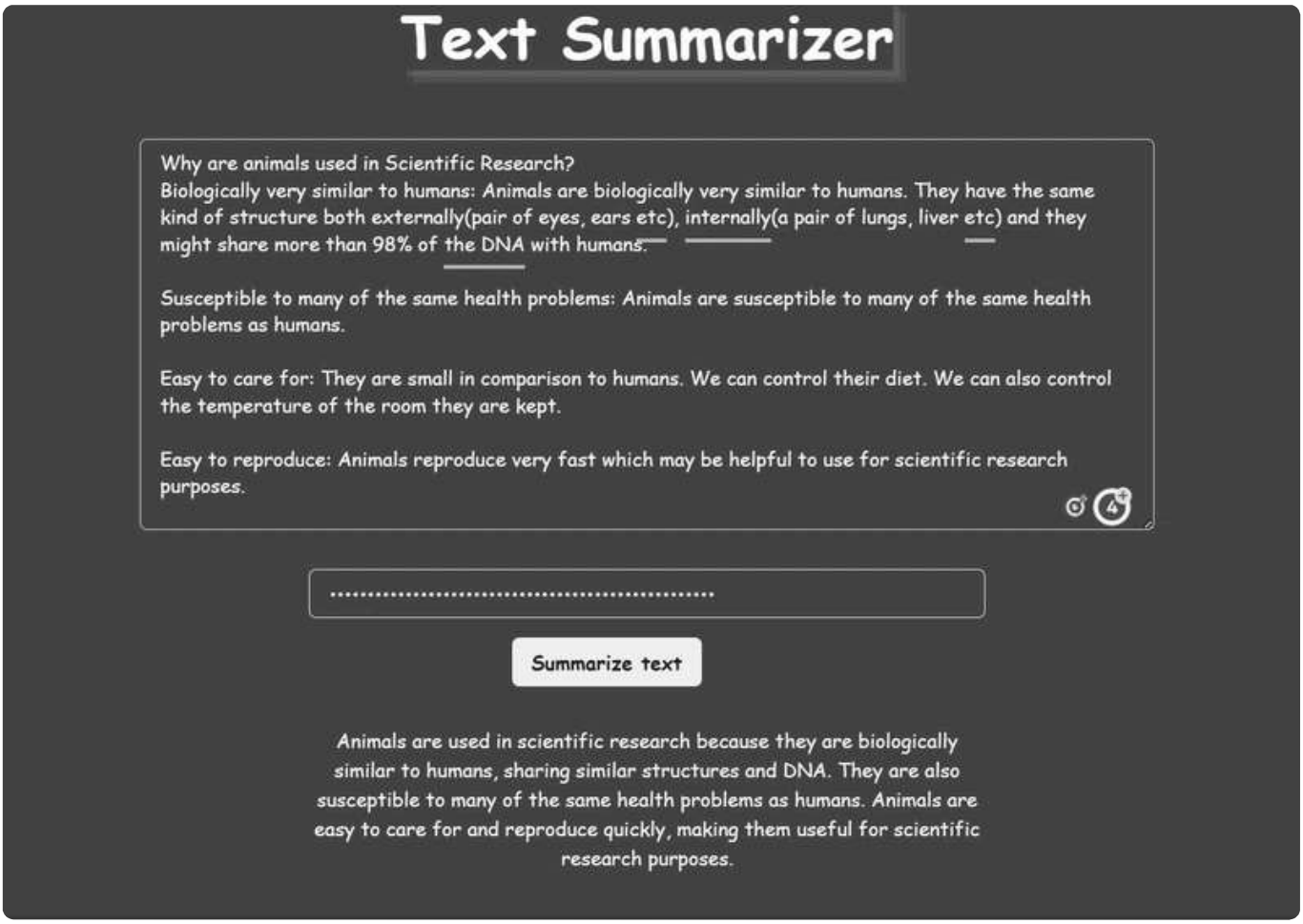
Build a Text Summarization app using Reflex (Pure Python)

#reflex #python #openai #machinelearning

Reflex is an open-source, full-stack Python framework that makes it easy to build and deploy web apps in minutes. You have most of the features of a frontend library like Reactjs and a backend framework like Django in one with ease in development and deployment. All while developing in a single language **PYTHON**.

We will use Reflex to build a text summarization app where a user will be able to input a text and the Openai llm and langchain will generate a summary of the text.

The following will be the output of the app:

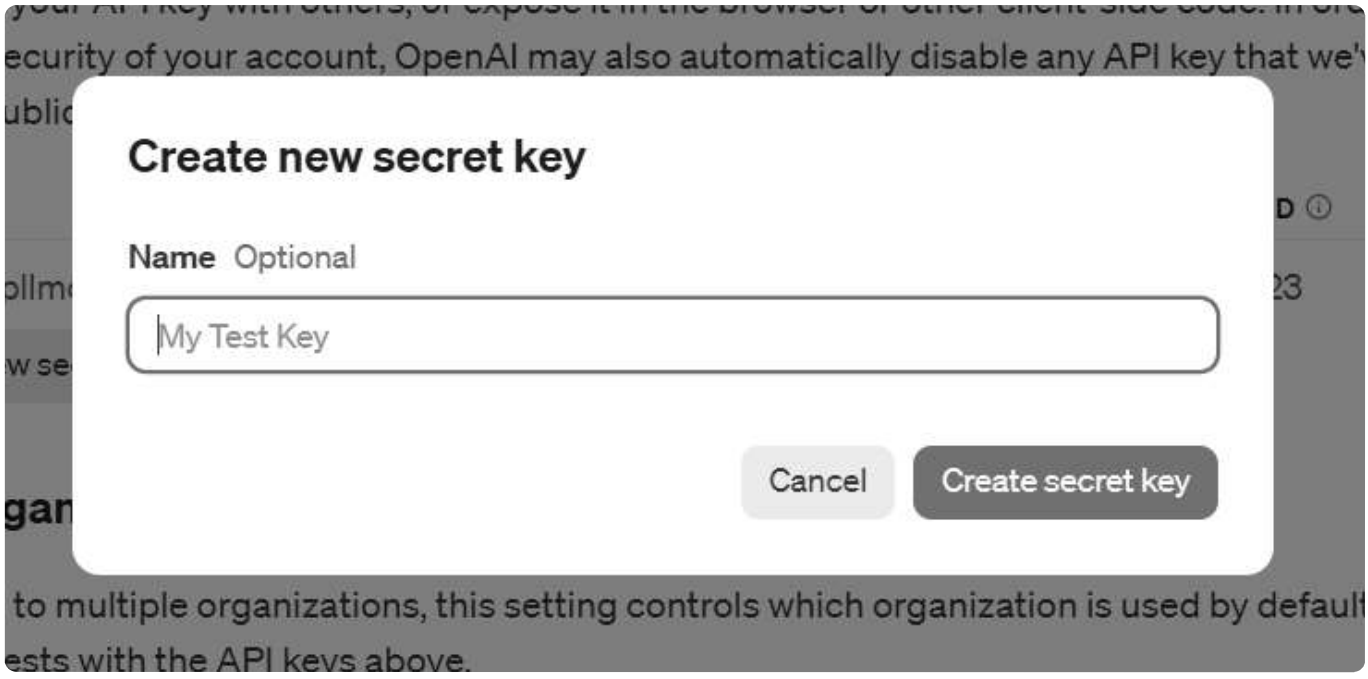


- Get an OpenAI API Key
- Create a new folder, open it with a code editor
- Create a virtual environment and activate
- Install requirements
- reflex setup
- text_summarizer.py
- state.py
- style.py
- .gitignore
- run app
- conclusion

Get an OpenAI API Key

First, get your own OpenAI API key:

- Go to https://platform.openai.com/account/api-keys.
- Click on the + Create new secret key button.
- Enter an identifier name (optional) and click on the Create secret key button.
- Copy the API key to be used in this tutorial



Create a new folder and open it with a code editor

Create a new folder and name it `text_summarizer` then open it with a code editor like VS Code.

Open the terminal. Use the following command to create a virtual environment `.venv` and activate it:

```
python3 -m venv .venv
```

```
source .venv/bin/activate
```

Install requirements

We will need to install `reflex` to build the app and also `openai tiktoken chromadb langchain` to generate the text summaries
Run the following command in the terminal:

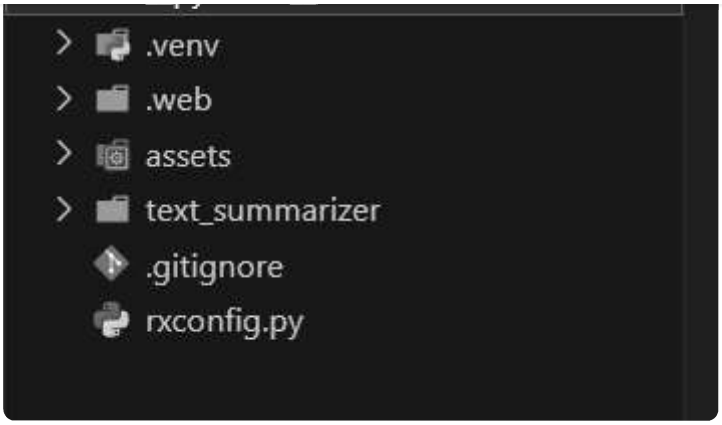
```
pip install reflex==0.2.9 openai==0.28.1 tiktoken==0.5.1 chromadb langchain==0.0.316
```

reflex setup

Now, we need to create the project using reflex. Run the following command to initialize the template app in `text_summarizer` directory.

```
reflex init
```

The above command will create the following file structure in `text_summarizer` directory:



You can run the app using the following command in your terminal to see a welcome page when you go to http://localhost:3000/ in your browser

```
reflex run
```

text_summarizer.py

We need to build the structure and interface of the app and add components. Go to the `text_summarizer` subdirectory and open the `text_summarizer.py` file. This is where we will add components to build the structure and interface of the app. Add the following code to it:

```
import reflex as rx

# import State and style
from text_summarizer.state import State
from text_summarizer import style


def full_text() -> rx.Component:
    """return a vertical component of heading and text_area."""
    return rx.vstack(
        rx.heading("Text Summarizer",style=style.topic_style),
        rx.text_area(
            value=State.large_text,
            placeholder="Enter your full text here",
            on_change=State.set_large_text,
```

```
    )


def openai_key_input() -> rx.Component:
    """return a password component"""
    return rx.password(
            value=State.openai_api_key,
            placeholder="Enter your openai key",
            on_change=State.set_openai_api_key,
            style=style.openai_input_style,
    )


def submit_button() -> rx.Component:
    """return a button."""
    return rx.button(
        "Summarize text",
        on_click=State.start_process,
        is_loading=State.is_loading,
        loading_text=State.loading_text,
        spinner_placement="start",
        style=style.submit_button_style,
    )


def summary_output() -> rx.Component:
    """return summary."""
    return rx.box(
            rx.text(State.summary, text_align="center"),
            style=style.summary_style,
    )


def index() -> rx.Component:
    """return a full_text, openai_key_input, submit_button, summary_output respectiv
    return rx.container(
        full_text(),
        openai_key_input(),
        submit_button(),
        summary_output(),
    )

# Add state and page to the app.
app = rx.App(style=style.style)
app.add_page(index)
app.compile()
```

The above code will render a text and the text area input, password input to enter your openai key, a submit button, and a box to show the summary

state.py

Create a new file `state.py` in the `text_summarizer` subdirectory and add the following code:

```
import reflex as rx

from langchain.chat_models import ChatOpenAI
from langchain.chains.summarize import load_summarize_chain
from langchain.docstore.document import Document


class State(rx.State):

    # The current large text to be summarized.
    large_text: str

    # openai key
    openai_api_key: str

    # the result
    summary: str

    is_loading: bool = False

    loading_text: str = ""



    def start_process(self):
        """Set state variables and summarize method."""
        self.is_loading = True
        self.loading_text = "generating summary...."

        return State.summarize
```

```
        llm = ChatOpenAI(temperature=0, model_name='gpt-3.5-turbo-16k', streaming=T

        docs = [Document(page_content=t) for t in self.large_text]

        # use load_summarize_chain to summarize the full text and return self.summar
        chain = load_summarize_chain(llm, chain_type="stuff")

        self.summary = chain.run(docs)
        self.summary
        yield

        # reset state variable again
        self.is_loading = False
        self.loading_text = ""
```

The above code uses `load_summarize_chain` with chain_type of "stuff" to generate the summary and send it to the front end. When the user clicks the "Summarize text" button, it triggers and calls the `start_process` method. `start_process` method assigns the `is_loading` argument of "Summarize text" button to `True` so that the button can start spinning with the text "generating summary...." to show that the app is generating the summary. `start_process` method then calls `summarize` method to generate the summary with the help of openai llm "gpt-3.5-turbo-16k" and yield the result or summary to the frontend. It then changes the `is_loading` argument of the button to `False` and `loading_text` to an empty string so as to return the button to its initial state.

style.py

Create a new file `style.py` in the `text_summarizer` subdirectory and add the following code. This will add styling to the page and components:

```
style = {
    "background-color": "#454545",
    "font_family": "Comic Sans MS",
    "font_size": "16px",
}

topic_style = {
```

```
    "font_weight": "bold",
    "box_shadow": "rgba(240, 46, 170, 0.4) 5px 5px, rgba(240, 46, 170, 0.3) 10px 10p
    "margin-bottom": "3rem",
}

textarea_style = {
    "color": "white",
    "width": "150%",
    "height": "20em",
}


openai_input_style = {
    "color": "white",
    "margin-top": "2rem",
    "margin-bottom": "1rem",
}


submit_button_style = {
    "margin-left": "30%",
}


summary_style = {
    "color": "white",
    "margin-top": "2rem",
}
```

.gitignore

You can add the .venv directory to the .gitignore file to get the following:

```
*.db
*.py[cod]
.web
__pycache__/
.venv/
```

Run app

Run the following in the terminal to start the app:

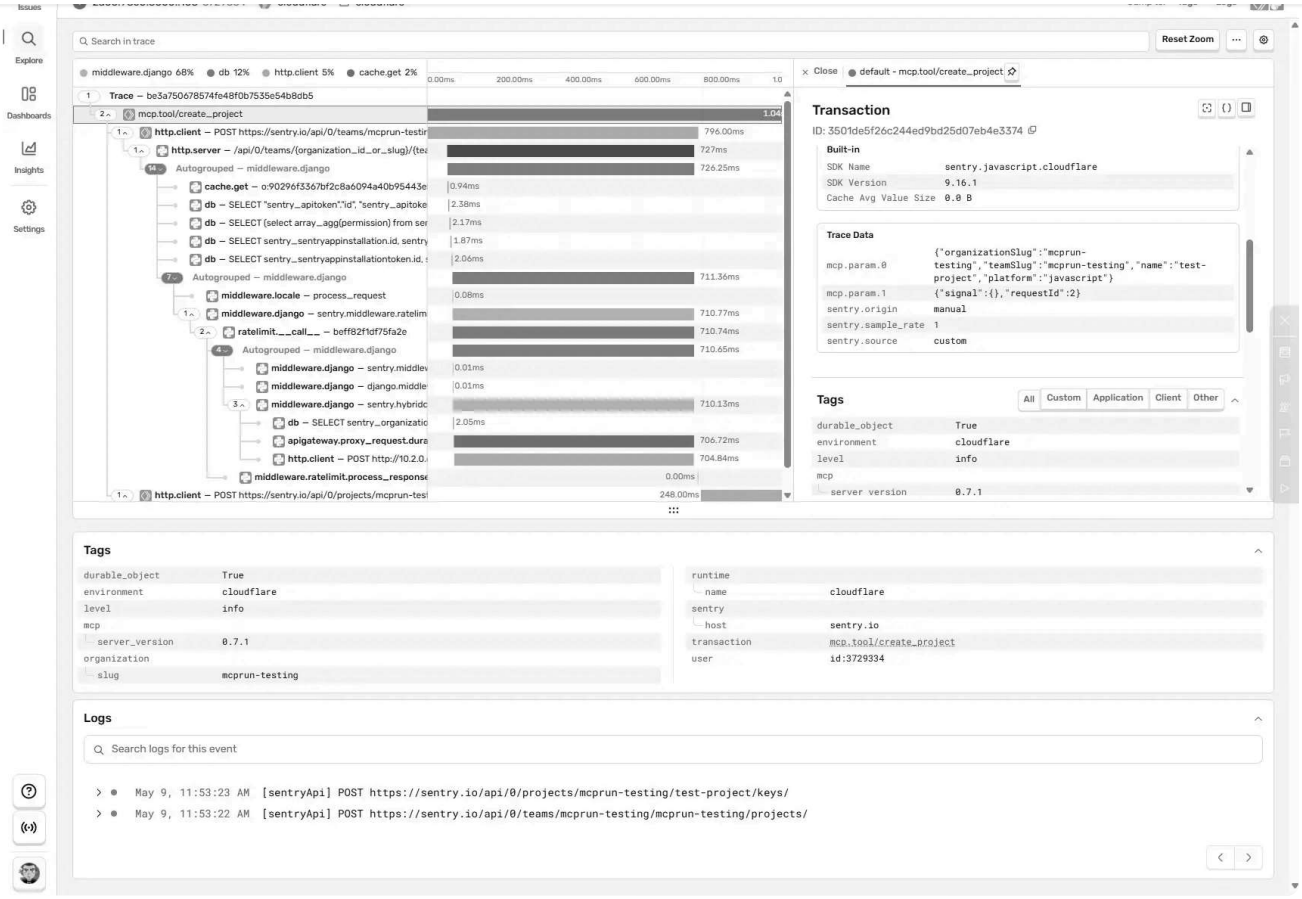
```
reflex run
```

You should see an interface as follows when you go to http://localhost:3000/

You can input the text that you want to be summarized. Also, enter your openai API key and then click the button to get your summarized text.

Conclusion

Reflex is awesome and a game-changer. You should try it out. You can get the code: https://github.com/emmakodes/text_summarizer.git



Monitoring your MCP Server in Production (with

Read More

Top comments (2)

artydev • 8 mai 24

Thank you :-)

Dan • 23 août 24

Emmanuel Onwuegbusi

I'm a Creative Software Engineer. I combine my experience as a Software engineer with Machine Learning skills to build data-driven applications.

WORK
Software Engineering

JOINED
29 mai 2023

Build Your Own AI Agent that Can Browse the Web and Take Actions 🤖

#programming #ai #python #tutorial

Build Your Own Podcast Producer AI Agent 🎙️

#ai #tutorial #webdev #python

Build Your Own AI People Research Agent using Supercog Agentic Framework 🔍

#ai #tutorial #python #programming