Name: Nalla Vamshi Krishna

Email: nvkrishna9396@gmail.com

Contact: +91 7793962832

5.1 Problem Statement

Explore the potential of large language model and Generative AI models like GPT for Natural Language Processing Tasks, such as text Generation, Summarization or Question Answering.

Approach

Model Selection:

This suggests that the model being used is likely Google AI's Gemini-Pro, a large language model (LLM) developed by Google AI. While the specific details of Gemini-Pro are not publicly available, it's likely a powerful LLM trained on a massive dataset of text and code, capable of generating different creative text formats, like poems, code, scripts, musical pieces, email, letters, etc. Model Extracted using google API key for Key link is give below

https://aistudio.google.com/app/u/1/apikey

i. Developed by: Google AI

ii. Model Type:
   a. Likely based on the Transformer architecture, a powerful deep learning model for natural language processing (NLP) tasks.
   b. Classified as a large language model (LLM), suggesting a complex architecture with a potentially high number of parameters (though the exact number is not publicly available).

iii. Functionality:
   a. Capable of generating different creative text formats based on the provided prompt, potentially including poems, code snippets, scripts, musical pieces, emails, letters, and more.

Libraries Used:

1. pipwin (Not Recommended):

    Function: pipwin is a package installer for Windows specifically. It's generally not recommended for Python package management on Windows.

    Alternatives: The preferred method for installing Python packages on Windows is using pip directly from the command prompt or terminal. You can activate a virtual environment first if you prefer to isolate project dependencies.

2. google-generativeai:

    Function: This library provides access to Google's Generative AI service, which allows you to interact with various large language models (LLMs) developed by Google AI.

    Functionality: You can use google-generativeai to send text prompts to LLMs and receive generated text in different formats based on the model's capabilities.

3. python-dotenv:

    Function: This library helps you load environment variables from a .env file. Environment variables are a way to store configuration settings outside of your code, making it easier to manage and keep sensitive information separate.

    Usage: You define variables in a .env file (e.g., API_KEY=your_api_key_here) and use python-dotenv to access them in your code. This keeps your API keys or other sensitive data out of your codebase.

4. streamlit:

    Function: This library allows you to create web applications in Python. Streamlit simplifies the process of building user interfaces (UIs) with minimal coding.

    Functionality: Streamlit provides components like text input fields, buttons, sliders, and more to build interactive dashboards or data visualization tools. It's great for prototyping or deploying simple web apps quickly.

5. os:

    Function: The os module provides functions for interacting with the operating system .

    Functionality: You can use os to access functionalities like listing files in a directory, creating directories, or getting environment variables (which can be useful for certain tasks).

6. logging:

Function: The logging module allows you to add logging capabilities to your Python code.

Functionality: You can use logging to record messages at different severity levels (e.g., DEBUG, INFO, WARNING, ERROR) to track the execution of your code, identify errors, or monitor performance.

7. Path:

Function: The Path class from the pathlib module provides a more object-oriented way to work with file paths.

Functionality: Path offers functionalities like joining paths, checking if a file exists, and manipulating file paths more conveniently compared to using raw strings.

8. find_packages and setup:

setup (from setuptools): Defines your Python package's metadata (name, version, etc.) and how it should be installed (dependencies, scripts). It's the core configuration file for distribution.

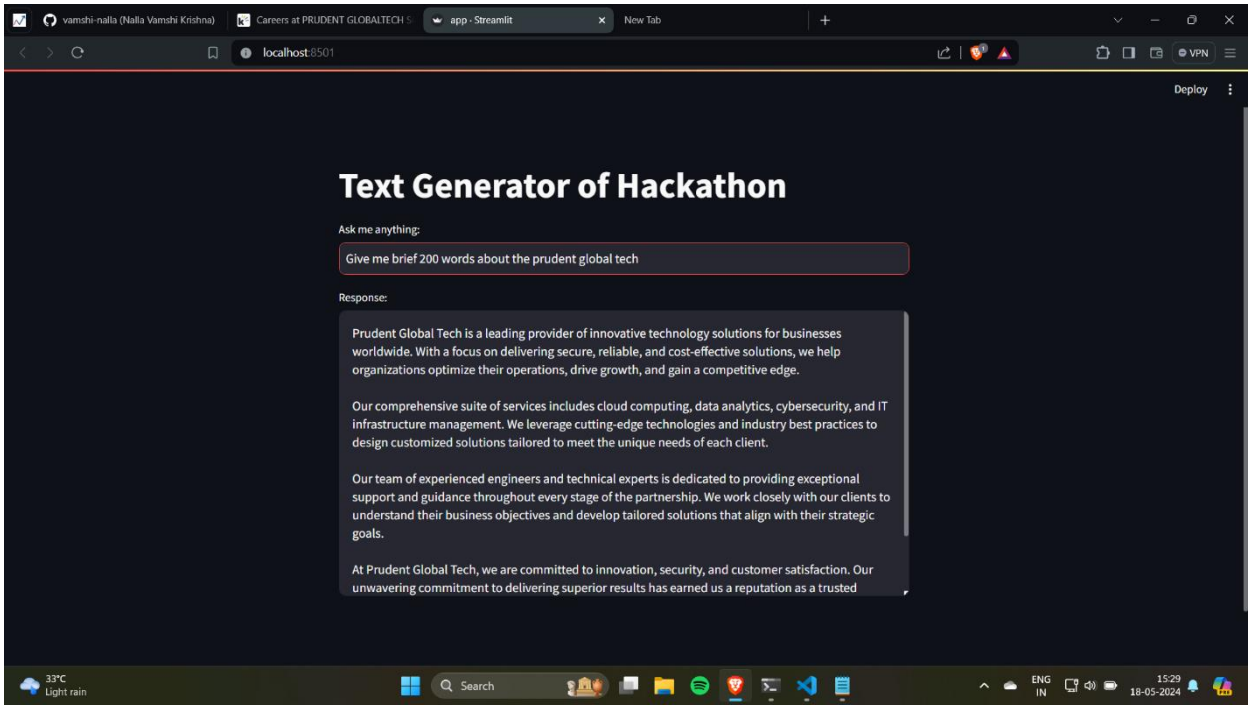
find_packages (from setuptools): Automatically finds all packages (directories with __init__.py) within your project, simplifying the packages argument in setup.

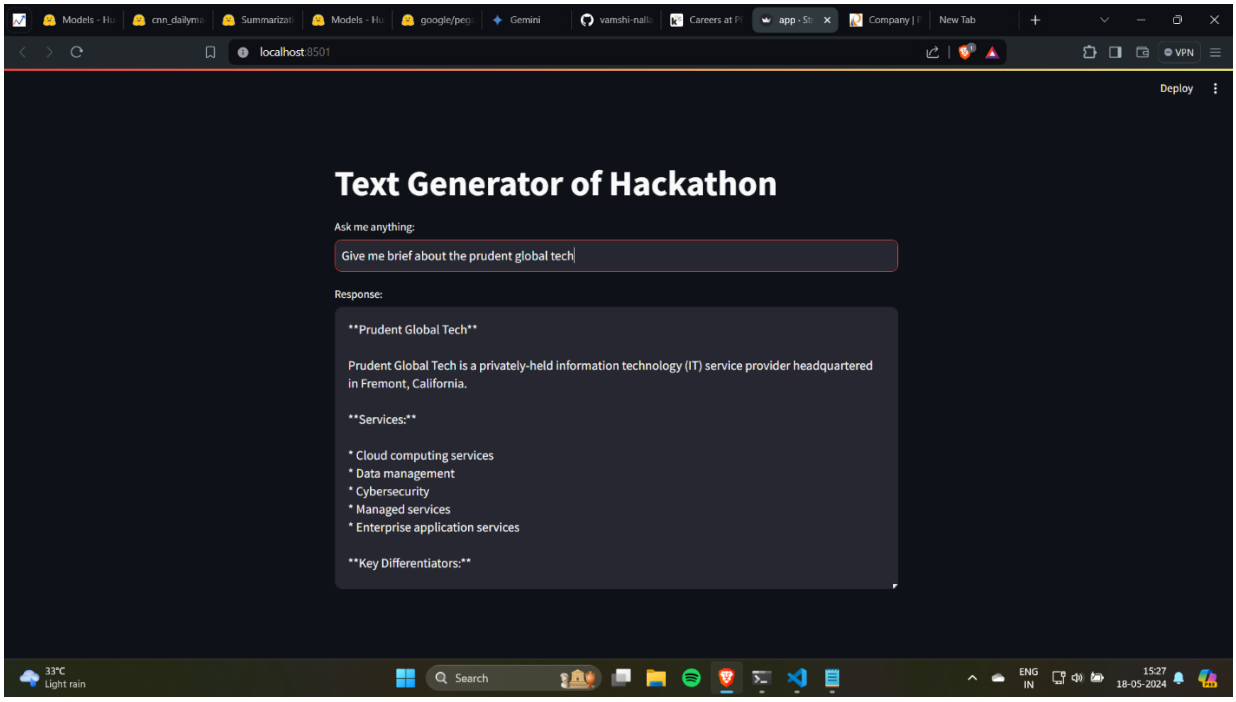
Together, they make creating distributable Python packages easier and more reliable.

Solution Architecture:

Model can easily Deployed on AWS and directly can be launched on EC2 instance without Dockerization. Model which we created don’t even use any custom data, LLM are pretrained with Large data and trained with billions of parameters

Screenshot Demonstrating the model's Output:





Source code and Deployment Steps:

Step - 1: Clone the repository or you can push the repository after the completion of project.

Step - 2: Create a conda environment after opening the repository and activate environment.

Step - 3: install the requirements.

Step - 4: Create a `.env` file in the root directory and add your GOOGLE_API_KEY credentials.

Step - 5: Try to Ignore the present virtual environment variables using gitignore.

Step - 6: Update the required files

Step - 7: Running the application on local machine

Note: For all the step commands are available in README.md file, I kindly request you please go through the GitHub link (https://github.com/vamshi-nalla/Prudent_Hackathon)

Thank You