DS/CS553 - Machine Learning Development and Operations (MLOps)

Group 2 - Jake Watson & Andrew Keane

Case Study 1 - Hugging Face, Gradio, and Git Familiarization

Hugging Face: <https://huggingface.co/spaces/cs-553-group2/case-study-1>

GitHub: <https://github.com/jakewatson91/cs553-group2>

Pitch Video: <https://www.youtube.com/watch?v=qK2Gcq-EcGo>

**Description of the product(s), their purpose, and target audience**

**Product Overview:**

* **"Ask the Greats" Chatbot**:
  + **Purpose**: This is an AI-powered chatbot designed to deliver famous motivational quotes or concise responses using text-generation models. It can either provide inspirational or theoretical quotes from renowned figures (books, movies, philosophers, business leaders) or switch to a more practical mode, offering actionable advice depending on the user's input. The chatbot allows users to interact with AI in a meaningful way, customizing the style and nature of responses through adjustable parameters such as **practicality**, **temperature**, and **top-p**.
  + **Target Audience**: The chatbot is aimed at a wide audience:
    - **Motivational Seekers**: Individuals looking for inspirational quotes or guidance from famous figures.
    - **AI Enthusiasts**: Users interested in experimenting with AI-generated content and customizing chatbot responses.
    - **Personal Development and Coaching**: Those in need of motivational advice or quotes for personal development.
    - **Educational Use**: It could also appeal to educators or students seeking famous quotes for writing, presentations, or general knowledge.
* **Gradio-Powered AI Interface**:
  + **Purpose**: This product uses Gradio to offer a user-friendly interface for interacting with the AI models. The interface allows users to adjust key settings, such as the chatbot’s response style (e.g., practical vs. theoretical), and provides a seamless experience for experimenting with various parameters like **temperature** (creativity in response generation), **top-p** (nucleus sampling), and **max tokens** (length of response). Users can choose between using a local model or accessing Hugging Face's API-based inference.
  + **Target Audience**:
    - **Developers and AI Practitioners**: Those who want to build or customize their own chatbot interfaces for text generation, as well as experiment with different AI models and their settings.
    - **Businesses**: Organizations looking for a customizable chatbot for internal or customer-facing motivational or instructional purposes.
    - **Non-technical Users**: The Gradio interface simplifies interacting with AI models, making it accessible even to those without technical expertise.

**Product Capabilities:**

* **Customizable Conversations**: Users can alter chatbot behavior by adjusting **practicality**, making the chatbot either provide actionable advice or focus on abstract, inspirational quotes.
* **Dynamic Response Generation**: With the flexibility to switch between a local model and Hugging Face API-based inference, the product provides scalable and dynamic response generation options based on user preferences.
* **User-Friendly Interface**: The Gradio-based interface ensures that users of all technical levels can easily interact with the AI by submitting queries and adjusting parameters.

**Potential Use Cases:**

* **Personal Development**: Users can receive motivational quotes to help improve daily motivation or gain insights from famous figures.
* **Learning and Reference**: People studying famous philosophical, business, or literary quotes can use the chatbot as a quick reference tool.
* **AI Experimentation**: Developers and AI enthusiasts can tweak parameters to test AI-generated responses and fine-tune output quality.

**Description of the models being used, including their architecture, purpose, and any datasets involved**

**Model Descriptions:**

1. **Hugging Face Zephyr-7B-beta Model**:
   * **Architecture**: Zephyr-7B-beta is a large language model (LLM) from Hugging Face with **7 billion parameters**. This model is based on the **transformer architecture**, which uses self-attention mechanisms to process input sequences in parallel. The architecture is optimized for natural language understanding and generation.
   * **Purpose**: This model is used for text generation, making it suitable for creating human-like text responses based on a given prompt. It can handle conversational tasks, text summarization, and creative writing.
   * **Dataset**: While the specific dataset for Zephyr-7B-beta is not explicitly detailed, models of this type are often trained on a variety of large, diverse text corpora, including web data (books, Wikipedia, articles, forums) to build general language understanding capabilities. These datasets may contain billions of text samples.
2. **Microsoft Phi-3-mini-4k-instruct**:
   * **Architecture**: The Phi-3-mini-4k-instruct model is another large-scale transformer-based model designed specifically for instruction-following tasks. Like Zephyr-7B, it uses a similar **transformer architecture**, but it is smaller in size and more optimized for scenarios requiring structured, clear instructions.
   * **Purpose**: Phi-3-mini-4k is designed to handle **instruction-based tasks**, such as generating text based on specific prompts, answering questions, and providing instructions or guidance. It is particularly effective for tasks that require understanding user queries and providing structured, helpful responses. In the chatbot, this model handles local inference for text generation.
   * **Dataset**: The Phi-3-mini-4k-instruct model is likely trained on a large, multi-domain dataset designed to handle a wide range of instruction-based tasks. This might include datasets such as **OpenAI's GPT-3 instruction-tuning datasets** or others with instructions and human feedback loops for better alignment with real-world tasks.

**Combined Function:**

Both models work together to provide efficient and versatile text generation. While the Zephyr-7B-beta model serves as the primary generator for inference through the Hugging Face API, the Phi-3-mini-4k model is used for local inference when external API access isn't utilized. The **transformer-based architecture** allows these models to excel in natural language tasks, generating coherent, contextually relevant, and highly adaptable responses.

Each model is capable of handling complex natural language understanding and generation tasks, making them highly effective for providing both conversational responses and more instructional or motivational outputs based on the chatbot's system configuration.

**Performance analysis of the products (e.g., how long do they take to answer, resource usage)**

Running the product/model numerous times and observing the resource usage supplied by Hugging Face behind the green "Running" button shows that the API version of the model requires about 6% of CPU and 1 GB of RAM. The execution is usually around 0.5 seconds. Occasionally, a very strange question could cause the response time to be around two seconds.

When running the model locally, the execution time is significantly more, around eight minutes. Comparing 8 minutes (Local) to 1 second (API), the performance ratio is 480:1, or approximately 500:1.

**Cost based analysis of the products (e.g., estimated cost if you have 1,000 users)**

Requiring 16 bytes per parameter, the model has approximately 60,000,000 parameters (1 GB / 16 bytes). If had 1,000 users, the resource requirements would be:

* CPU = 60 CPUs
* Memory = 1,000 GB

Considering options:

* AWS Intel Sapphire Rapids w/8 CPUs and 16 GB RAM, $0.27/hour
  + Would need eight of these to get enough CPUs
  + Would need 62 of these to get enough RAM
  + Cost = $0.27/hour \* 62 instances \* 1/3,600 \* 1000 users = **$4.65/hour**
* NVIDIA L4 w/GPUs and 96 GB RAM, $3.80/hour
  + Would need 11 of these to get enough RAM
  + Cost = $3.80/hour \* 11 instances \* 1/3,600 \* 1000 users = **$11.61/hour**

The cost is $4 to $12/hour. Using $10/hour, the cost would then be:

* Cost = $10/hour \* 24 hours \* 365 days =~ **$90,000/year**

Yes, that DOES get expensive!

**Comments and/or concerns, such as potential security issues, data privacy, and scalability**

The models used for this product are static, pre-trained models. They do not learn from the questions input by users, so any user data is not stored beyond the time required to generate each response. This means that the security/privacy concerns are near zero.

As for scalability, the resources required would scale linearly based on the number of users. Theoretically, the product could be scaled to almost any size with enough money and GPUs.

**Additional insights, challenges faced, and potential future improvements**

There were numerous issues related to the environment that slowed progress.

* The only way to allow two users to share a Hugging Face space was to create a Hugging Face organization, grant users access to the organization, and then put the space inside of the organization. For Hugging Face organizations, both users could be full admins.
* For GitHub, a second users could be added as a repository (repo) collaborator. Though the second user could not be made an equal admin. Therefore, there was no way for the second user to access/modify repo settings.
* Deploying to Huggingface was initially an issue. The working .yml file included the following line: git push https://jakewatson91:${{ secrets.HF\_TOKEN }}@huggingface.co/spaces/cs-553-group2/case-study-1 main
* The system prompt went through several iterations to produce the best results. Clear instructions were provided to limit answers to three direct quotes with no extra “fluff.” This proved difficult as LLM’s seem to always want to add an explanation. The final base system message was: “You are a chatbot that responds with famous quotes from books, movies, philosophers, and business leaders. Provide no advice, commentary, or additional context. Your responses should be concise, no more than 3 quotes, and consist only of famous motivational quotes."
* The "Use Local Model" button may not be working. Running the local version in Hugging Face is producing the same performance results (API performance). But running in GitHub produced the timing for the Local run (eight minutes). Looking into this could be a future improvement.
* Adding the practicality slider proved more difficult than expected and produced mixed results. The goal was to have a practicality message append to the predefined system message when the value was changed (ie. if practicality > 0.5, append: “provide actionable advice or direct instructions”).

**Appendix - AI Prompts Used**

Some code and documentation were initially generated by ChatGPT and then modified to correct errors, fix focus, update formatting, etc.

Here are the prompts used:

**Create Unit Test**

* Create one simple unit test that will be run as a GitHub Action for the following code. <code for app.py>

**Product Overview**

* For the same code, provide a description of the product(s), their purpose, and target audience.

**Model Overview**

* For the same code, provide a description of the models being used, including their architecture, purpose, and any datasets involved.