NLP COMM061 Group Coursework

Sourav Sen, Anisha Rajesh, Manan Taneja, Lyuijia Qian, Atharva Mahesh Sapkal

May 24, 2024

1 Q1. Model Serving Survey

In recent years that there has been an massive increase in the number of model serving options. In order to compare there offerings we borrow dimesnions from a typical Model Development Lifecycle

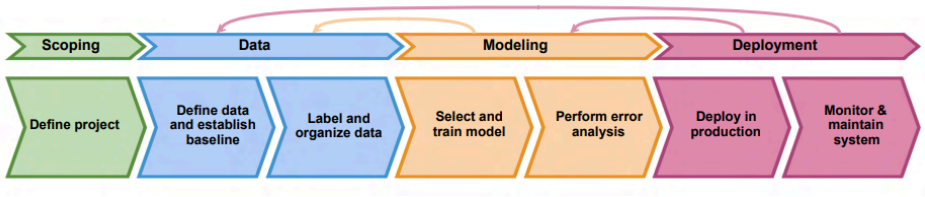

Figure 1: Model Development Lifecycle

Here as we can see the process of defining AI solutions, building, deploying and monitoring is essentially a feedback cycle and therefore it matters that we have a coherent set of tools to be able to iterate over this cycle.

The Big Tech Cloud hosting players AWS, GCP, Asure all have elaborate set of tools and frameworks for supporting these MDLC. However there is a large ecosystem of smaller players whos combined offering can provide a great DevX experience for AI model building for free or significantly lower costs. For our survey we compare the landscape of offerings in terms of 5 major criteria

- Hosting Platform - Google, AWS, Azure, Heroku, HuggingFace,
- Application Framework eg Flask, StreamLit, Gradio , FastAPI
- Metrics Monitoring eg Weights & Biases, Cometml
- CI/CD options eg GitHub integration , Vendor specific
- Free/Paid Resourcing eg CPU/GPU/RAM

Table 1 summarises a subset of the offerings . Note this is by no means a comprehensive survey as the table in the next page can easily be extended by another 50 offerings. Some notable exclusions here are

- Neptune.ai
- Comet.ml
- H2O

Amongst these Comet and Neptune is free for academics. They both offer training hours and storage of experimental results ahdv overlapping capabilities with Weights & Biases.

1.1 Selection Rationale

We wanted to choose a setup that is completely free and no major additional learning curve. We ruled out AWS , GCP and Azure because while they do have free tiers , it is easy to fall foul of the fine print and can incur accidental charges. these are great platform in a professional setting where there are dedicated Fin-Ops safeguards in place.

On the other hand the Hugging Face ecosystem has been going from strength to strength in recent years and addresses several of our requirements.

- The HF Hub acts as Model registry that is widely used and allows model comparision
- Easy integration with GitHub Actions to deploy models
- Metrics Monitoring via Weights and Biases

Hugging Face Spaces offers a paid Inference Endpoint solution that costs as low as $0.032 per CPU corehr and $0.5 per GPUhr W&B will detect if your dataset has changed and only update the dataset version is different.

We also compare direct Streamlit hosting vs Hugging Face running Streamlit and Gradio servers. Streamlit Hosted solutions are not great for any kind of API testing as they are meant for Web UI based on solutions only . Instead we relied more on the Gradio + FastAPI based hosting on Hugging Face API to move both web and API testing for our hosted model.

A shortcoming of our choice compared to the major cloud providers is that one of the key aspect of Model deployment is A/B testing where we can run 2 slightly differnt models in production for a particular AI problem and let it serve randomly or by design different sets of incoming requests. Then we can monitor and compare and expand the rollout of different models. Hosting models in HuggingFace or Streamlit would require slightly manual activities to enable this

ID	Hosting Platform	Framework	Resources (CPU/GPU/RAM)	Metrics	
1	Hugging Face Spaces	Gradio	2 vCPUs / No GPU / 8 GB	GitHub Actions	
2		Streamlit	2 vCPUs / No GPU / 8 GB	GitHub Actions	
3		Flask	2 vCPUs / No GPU / 8 GB	GitHub Actions	
4	Google Colab	Gradio	2 vCPUs / 1 GPU / 12 GB	GitHub Actions, Custom Scripts	
5		Streamlit	2 vCPUs / 1 GPU / 12 GB	GitHub Actions, Custom Scripts	
6		Flask	2 vCPUs / 1 GPU / 12 GB	GitHub Actions, Custom Scripts	
7	Heroku	Gradio	1-4 vCPUs / No GPU / 512MB-8GB	GitHub Actions, CircleCI, Travis CI, Heroku CI/CD	
8		Streamlit	1-4 vCPUs / No GPU / 512MB-8GB	GitHub Actions, CircleCI, Travis CI, Heroku CI/CD	
9		Flask	1-4 vCPUs / No GPU / 512MB-8GB	GitHub Actions, CircleCI, Travis CI, Heroku CI/CD	
10	AWS EC2	Gradio	Custom	AWS CodePipeline, Jenkins, GitHub Actions, GitLab CI/CD, CircleCI	Highly s
11		Streamlit	Custom	AWS CodePipeline, Jenkins, GitHub Actions, GitLab CI/CD, CircleCI	
12		Flask	Custom	AWS CodePipeline, Jenkins, GitHub Actions, GitLab CI/CD, CircleCI	
13	Google Cloud Run	Gradio	2 vCPUs / No GPU / 2 GB	Google Cloud Build, Jenkins, GitHub Actions, GitLab CI/CD, CircleCI	
14		Streamlit	2 vCPUs / No GPU / 2 GB	Google Cloud Build, Jenkins, GitHub Actions, GitLab CI/CD, CircleCI	
15		Flask	2 vCPUs / No GPU / 2 GB	Google Cloud Build, Jenkins, GitHub Actions, GitLab CI/CD, CircleCI	
16	Azure App Service	Gradio	1-4 vCPUs / No GPU / 1-14 GB	Azure DevOps, GitHub Actions, Jenkins, CircleCI	
17		Streamlit	1-4 vCPUs / No GPU / 1-14 GB	Azure DevOps, GitHub Actions, Jenkins, CircleCI	
18		Flask	1-4 vCPUs / No GPU / 1-14 GB	Azure DevOps, GitHub Actions, Jenkins, CircleCI	
19	IBM Cloud Foundry	Gradio	1-4 vCPUs / No GPU / 512MB-32GB	IBM Cloud Continuous Delivery, Tekton, Jenkins, GitHub Actions	
20		Streamlit	1-4 vCPUs / No GPU / 512MB-32GB	IBM Cloud Continuous Delivery, Tekton, Jenkins, GitHub Actions	
21		Flask	1-4 vCPUs / No GPU / 512MB-32GB	IBM Cloud Continuous Delivery, Tekton, Jenkins, GitHub Actions	

Table 1: Hosting Platform Comparison for Gradio, Streamlit, and Flask Applications

Codebase	Purpose	Framework	Comments
HF::/bert-cased-plodcw-sourav	Storage of Fine-tuned PLOD-CW Models	Trained from Heron with Jupyter	Needs a GitHub Action driven training setup
GH::/nlp-cw-group27	Streamlit Community Deployment	Streamlit	However, this has the problem that Streamlit doesn't natively offer a way to be used as an API server; in particular, it doesn't allow us to set HTTP headers to application/json
GH::/nlp-cw-group27	AWS Deployment	Flask	TBD
HF::/hf_gradio_plodcw_group27	HF Space Gradio + FastAPI deployment	Gradio	This hosts a Gradio Web UI as well as a FastAPI server **
HF::/hf-nlp-cw-group27	HF Space Streamlit deployment	Streamlit	This only offers a Streamlit Web UI

Table 2: Various deployments for Gradio, Streamlit, and Flask Applications ** Option is used for testing further

2 Q2. Web Service Implementation

We iterated over a few different implementations as shows in Table 2
Our Hugging Face StreamLit URL is HF Streamlit
Our Hugging Face Gradio + FastAPI URL is HF Gradio
Our implementation is for the webservice is in GitHub repo

- `app_flask_plodcw.py` - Flask implementation
- `app_streamlit_plodcw.py` - Streamlit implementation
- `testing_plodcw_deployment.ipynb` - Endpoint Testing and Stress Testing Codes
- `README.md` - describes how to deploy to StreamList Hosting, HuggingFace Space

Both applications loads the dataset of PLOD_CW , so it can offer some ready samples to try out , has the ability to request a NER token classification and then renders the output somewhat alike how `displacy` does it There is also a dropdown to toggle between our trained model and other baseline models and see the results.

We also have a link to show the runtime performance of our application . We have a few variations here

1. Train in Heron (Jupyter or command line) , Post Model to HF, Host in Streamlit Cloud to use HF model ¡ur¿ . Note Streamlit offers only 1 GB memory
2. Train in Heron (Jupyter or command line) , Post Model to HF, Host in HuggingFace Space ¡url¿
3. Train in Heron (Jupyter or command line) , Post Model to HF and WandB, Host in Streamlit Cloud. ¡url¿ This time we can use some cool WandB metrics and NER visualization out of the box

One of the lessons here has been that HF Models store classes as LABEL_1 , LABEL_2 etc and so its the responsibility of the model serving app to map these classes back into the label space of the problem eg BO etc

3 Q3. Endpoint Testing

As mentioned before `ner_service_testing.ipynb` is the notebook that is used to do basic Endpoint Testing. The model that was uploaded to https://huggingface.co/LampOfSocrates/bert-cased-plodcw-sourav has the following confusion matrix on the test dataset . ie ¿ 92% f1 score. The hosted

A key aspect that has not been covered in the course is the aspect of Authentication , Authorization and DDOS checks and Penetrating Testing .

What can be better : Our endpoint ought to be collecting user feedback and so a user can point out when the model has got it wrong. This might help with the human labelling problem

4 Q4. Stress Testing

In order to simulate solid performance testing, the clients of any hosted model service ought to be distributed clients This simulates real peak load for a service. There are several industrial scale testing solutions in this space. A home grown solution around the python package '**locust**' has been used here which simulates N concurrent users making simultaneous requests to the the HF hosted service and then captures the results. Note It is not sufficient to conclude anyting from this information , but the exercise of distributed testing has been educational .

We find that there is a median response time of 110 seconds between Heron labs and Hugginf Face Spaces with very small variability. This shrinks to 50ms when tested from local machines.

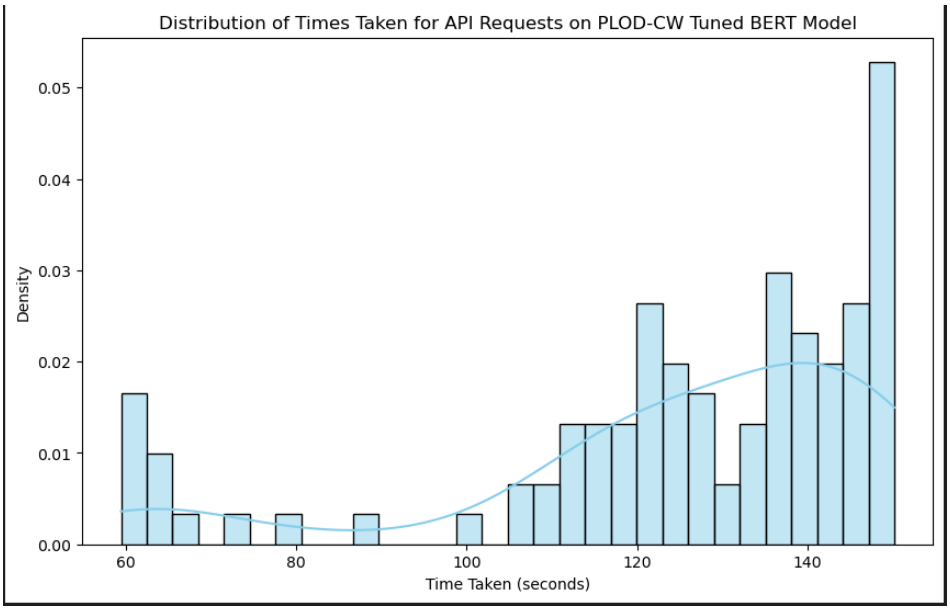

Figure 2: Latency Distribution of predictions from Deployed Service

Important Lesson

Key Takeaway: Realistic stress testing requires a distributed farm of clients. Here by using a single client machine and multiple threads our latency scores can be affected by the load and I/O pressures on the host machine ie personal laptop or Heron

5 Q5. Monitoring

Here we want to capture user inputs and model predictions and be able to store this in the form a log file. In all our applications we use the python logging module to store a local log . We also offer an end point /logs to stream these logs back to a Jupyter notebook `testing_plodcw_deployment.ipynb` contains the details

6 Q6. CI/CD Pipelines

We rely on GitHub Actions to implement our CI/CD and upload our trained model to HF and WandB

* Training Time For every experiment run on Jupyter Notebooks, we automatically upload to W&B to capture the training metrics.

7 Recordings

Note that there has been scheduling conflicts within the group given that this submission coincides with several other submissions and exams. And so there is multiple separate recordings related to this submission.

8 Appendix