Lab 0 - Deploy Llama2 and Embedding Models

Important
Run with **JupyterLab**: We're going to use `lab-00-setup/Lab_0_Warm_Up_Deploy_EmbeddingModel_Llama2_on_Nvidia.ipynb` notebook for this section

Llama 2 Model

(generated by a StableDiffusion Model)

Overview

Meta developed and publicly released the Llama 2 family of large language models (LLMs), a collection of pretrained and fine-tuned generative text models ranging in scale from 7 billion to 70 billion parameters. Our fine-tuned LLMs, called Llama-2-Chat, are optimized for dialogue use cases. Llama-2-Chat models outperform open-source chat models on most benchmarks we tested, and in our human evaluations for helpfulness and safety, are on par with some popular closed-source models like ChatGPT and PaLM. We provide a detailed description of our approach to fine-tuning and safety improvements of Llama-2-Chat in order to enable the community to build on our work and contribute to the responsible development of LLMs.

Model Developers Meta AI

Variations Llama 2 comes in a range of parameter sizes — 7B, 13B, and 70B — as well as pretrained and fine-tuned variations.

Input Models input text only.

Output Models generate text only.

Model Architecture Llama 2 is an auto-regressive language optimized transformer. The tuned versions use supervised fine-tuning (SFT) and reinforcement learning with human feedback (RLHF) to align to human preferences for helpfulness and safety.

	Training Data	Params	Content Length	GQA	Tokens	LR
Llama 2	A new mix of publicly available online data	7B	4k	✗	2.0T	3.0 x 10-4
Llama 2	A new mix of publicly available online data	13B	4k	✗	2.0T	3.0 x 10-4
Llama 2	A new mix of publicly available online data	70B	4k	✓	2.0T	1.5 x 10-4

Llama 2 family of models. Token counts refer to pre-training data only. All models are trained with a global batch-size of 4M tokens. Bigger models - 70B -- use Grouped-Query Attention (GQA) for improved inference scalability.

Model License Information

Use of this model is governed by the Meta license available at https://ai.meta.com/resources/models-and-libraries/llama-downloads/

Host Llama 2 Model as a SageMaker Endpoint

There are 2 ways to host a Llama2 model on SageMaker,

1. JumpStart UI - Click and Deploy
2. Notebook and sagemaker python SDK

We're going to use Option 2. for this workshop.

To deploy model, please ensure you have accepted Llama 2 license using Jupyter notebook ipywidget dropdown,

```
LLAMA2_EULA = False

from ipywidgets import Dropdown

eula_dropdown = Dropdown(
```

```
    options=["True", "False"],
    value="False",
    description="**Please accept Llama2 EULA to continue:**",
    style={"description_width": "initial"},
    layout={"width": "max-content"},
)
display(eula_dropdown)
```

Please accept Llama2 EULA to continue: True

```
LLAMA2_EULA = eval(eula_dropdown.value)
print(f"User set EULA to --> {LLAMA2_EULA}")
```

```
User set EULA to --> True
```

Then, we instantiate a new JumpStart model for Llama 2 model using the chat model id `meta-textgeneration-llama-2-<size>-f` and model version `*` (latest).

```
JUMPSTART_SRC_MODEL_NAME = "meta-textgeneration-llama-2-7b-f"
MODEL_VERSION = "*"
INSTANCE_TYPE = "ml.g5.2xlarge"
MODEL_NAME = "meta-llama2-7b-chat-tg-model"
ENDPOINT_NAME = "meta-llama2-7b-chat-tg-ep"
```

```
llama2_13b_model = JumpStartModel(
    model_id=TG_JUMPSTART_SRC_MODEL_NAME,
    role=role,
    name=TG_MODEL_NAME
)
```

Now deploy!

```
%%time
print("===== Llama2 SageMaker Deployment =====")

print("\nPreparing to deploy the model...")
llama2_13b_model.deploy(
    endpoint_name=TG_ENDPOINT_NAME,
    instance_type=TG_INSTANCE_TYPE,
    accept_eula=LLAMA2_EULA
)
print("\n===== Llama2 Deployment Complete =====")
```

> **Important**
> Llama2 model deployment takes between 5-8 minutes. Llama2 Endpoint should be `In Service` before using the model for inference

Embedding Model

The purpose of deploying an embedding model is to leverage the embedding model to generate text embedding in order to build a Retrieval Augmented Generation (RAG) application (in Lab 3).

Overview

This is a transformer-based model from Hugging Face without a text generation model head. It takes a text string as input and produces an embedding vector with 384 dimensions. While the published transformer-based model produces an embedding for every token in the input sequence, this model uses mean pooling, or an element-wise average, to aggregate these token embeddings into a sequence embedding.

The deployed model can be used for running inference on any input text. Example python code for how to run inference on the deployed model is given in a notebook you can open by clicking 'Open Notebook' on the model endpoint page that is created when deploying the model.

For a provided list of text inputs, the model outputs a list of 384-dimensional embeddings as a list of floats. Below are two example inputs with the first 5 elements of the embedding vector:

```
Example-1
Input Text: How cute your dog is!
Embedding Head: [-0.01781977154314518, -0.015948351472616196, 0.028173338621854782, 0.046047914773225784, -0.0474776066839695]

Example-2
Input Text: The mitochondria is the powerhouse of the cell.
Embedding Head: [-0.03643149882555008, 0.04998566210269928, -0.06370298564434052, 0.04446597024798393, -0.022807106375694275]
```

Host Embedding Model as a SageMaker Endpoint

Similar to Llama 2, we're going to leverage JumpStart to instantiate a new All MiniLM L6 v2 embedding model using model id `huggingface-textembedding-all-MiniLM-L6-v2` and model version `*` (latest).

```
EMB_JUMPSTART_SRC_MODEL_NAME = "huggingface-textembedding-all-MiniLM-L6-v2"
EMB_INSTANCE_TYPE = "ml.g5.2xlarge"
EMB_MODEL_NAME = "hf-allminil6v2-embedding-model"
EMB_ENDPOINT_NAME = "hf-allminil6v2-embedding-ep"
```

```
allminiv2_l6_model = JumpStartModel(
    model_id=EMB_JUMPSTART_SRC_MODEL_NAME,
    role=role,
    name=EMB_MODEL_NAME
)
```

Now deploy!

```
%%time
print("===== EmbeddingModel SageMaker Deployment =====")

print("\nPreparing to deploy the model...")
allminiv2_l6_model.deploy(
    endpoint_name=EMB_ENDPOINT_NAME,
    instance_type=EMB_INSTANCE_TYPE,
)
print("\n===== EmbeddingModel Deployment Complete =====")
```

Important
Embedding model deployment takes between 3-6 minutes. Embedding Model Endpoint should be `In Service` before using the model for inference

Deployment Complete!
Congratulations, You've successfully hosted LLMs!

Previous Next