Title: Fine-Tuning and Evaluation of Large Language Models with Amazon SageMaker

Name - **Rishu Raj Nayak**
Roll No - **21051840**

Introduction:

Large Language Models (LLMs) have revolutionized Natural Language Processing (NLP). Hugging Face and Amazon SageMaker provide powerful toolkits for fine-tuning and deploying these LLMs. This report explores a code example that demonstrates how to:

1. Fine-tune open LLMs using Hugging Face Transformers and the trl library for task-specific performance.
2. Deploy and evaluate a fine-tuned LLM on Amazon SageMaker for real-world inference.

Code Breakdown:

Section 1: Environment Setup:

- Libraries: Import necessary libraries (transformers, datasets, sagemaker, huggingface_hub).
- IAM Setup: Configure authentication and permissions for interacting with AWS services, specifically SageMaker.
- Environment Variables: Set SageMaker session bucket, execution role, and region.

Section 2: Dataset Preparation:

- Dataset: Utilize the sql-create-context dataset containing natural language instructions, database schemas, and corresponding SQL queries.
- Conversational Format: Convert the dataset into conversational format, suitable for fine-tuning LLMs in the context of text-to-SQL tasks.
- Dataset Splitting: Partition the dataset into training (80%) and testing sets (20%) for model training and evaluation, respectively.
- S3 Upload: Upload formatted datasets to Amazon S3 using an integration for efficiency and simplified access during training.

Section 3: LLM Fine-tuning:

- SFTTrainer: Employ the trl library's SFTTrainer to streamline the fine-tuning process. It inherits advantages of the Hugging Face Trainer.
- run_sft.py: Leverage a custom script to execute fine-tuning; allows parameterization and adjustment.
- Hyperparameters: Configure parameters for model (codellama/CodeLlama-7b-hf),

training process (epochs, batch size, etc.), and optimization (QLora, FlashAttention2).
- Hugging Face Estimator: Package the code and hyperparameters into an "estimator" for training job execution on SageMaker.
- SageMaker Training Job: Initiate a training instance using a specified instance type, resource limits, and the estimator, which then executes run_sft.py on the training data.

Section 4: Model Deployment and Evaluation:

- Model Artifacts: Retrieve the S3 location of the fine-tuned model after successful training.
- Inference Container: Utilize a Hugging Face Inference DLC, optimized for deploying and serving LLMs.
- Configuration: Specify parameters like instance type, GPU utilization, and input/output length limits for inference.
- Hugging Face Model: Create a HuggingFaceModel for encapsulating the fine-tuned model, container image, and configuration.
- Deployment: Deploy the model to an Amazon SageMaker endpoint for real-time text-to-SQL generation.
- Evaluation: Load the testing data, construct prompts, and evaluate the model on a subset of the test set to determine accuracy.

Key Considerations:

- Cost-Efficiency: Amazon SageMaker facilitates cost-effective on-demand instance usage for training and deployment.
- Dataset Quality: LLMs benefit from high-quality task-specific datasets (e.g., curated vs. synthetically generated).
- Evaluation: Careful evaluation metrics specific to text-to-SQL (such as query execution accuracy against databases) are essential for assessing practical performance.

Conclusion:
This code demonstrates a practical example of adapting and evaluating powerful LLMs like CodeLlama on Amazon SageMaker. This approach is applicable to various NLP tasks, enabling refined models for real-world applications.