Lab 5 - Foundation Model Evaluation

> **Important**
> Run with **JupyterLab**: We're going to use scripts inside `lab-05-llm-eval/Lab_5_FM_Eval_on_SageMaker_Studio.ipynb` notebook for this section

Contents

Overview

Foundation Model Evaluation (FMEval) refers to the systematic assessment of large language models, such as `Llama2`, `Amazon Titan` or `Anthropic Claude-v2`, on various criteria including accuracy, fluency, comprehensiveness, bias, and ethical considerations. FMEval is crucial because it determines the model's effectiveness in understanding and generating human-like text, its ability to provide reliable and unbiased information, and its adherence to ethical guidelines. For users, particularly those in business or research, this evaluation is key in deciding the best model for production use. It ensures the chosen model aligns with their specific needs and ethical standards, whether it's for customer service automation, content creation, or data analysis. By thoroughly evaluating these models, users can optimize their performance, mitigate risks associated with incorrect or unethical outputs, and ensure the model's outputs are beneficial and safe for their intended audience.

(image generated by stable-diffusion)

Foundation Model Evaluation

Foundation Models (FM) can be evaluate for a wide range of tasks with different intended purposes. Below are some popular FM evaluation categories to score a model,

1. **Open-ended Generation:** Open-ended text generation is a foundation model task that generates natural language responses to prompts that don't have a pre-defined structure, such as general-purpose queries to a chatbot.
2. **Text Summarization:** Text summarization is used for tasks, such as creating summaries of news, legal documents, academic papers, content previews, and content curation. The following can influence the quality of responses: ambiguity, coherence, bias, fluency of the text used to train the foundation model, and information loss, accuracy, relevance, or context mismatch.
3. **Question & Answering:** Question answering is used for tasks such as generating automatic help-desk responses, information retrieval, and e-learning.
4. **Classification:** Classification is used to categorize text into pre-defined categories. Applications that use text classification include content recommendation, spam detection, language identification and trend analysis on social media. Imbalanced, ambiguous, noisy data, bias in labeling are some issues that can cause errors in classification.
5. **Human-in-the-loop evaluations:** Human in the loop (HIL) FMEval is used for a wide variety of LLM tasks, including the tasks listed above. HIL eval can be a bit tedious but could provide the most comprehensive evaluation of models that emulates how a model might perform in a real-world scenario. SageMaker Ground Truth can help with HIL FMEval

To understand what each of these mean in depth, please refer to: https://docs.aws.amazon.com/sagemaker/latest/dg/clarify-foundation-model-evaluate-overview.html.

SageMaker FMEval

SageMaker Clarify offers Foundation Model Evaluations (FMEval) as a single place to evaluate and compare model quality and responsibility metrics for any language model (LM). Use FMEval to evaluate pre-trained and fine-tuned language models, text classifiers, and more. A foundation model serves as a starting point, from which you can develop downstream natural language processing (NLP) applications. FMEval provides connectors to pre-trained, text-based foundation models from SageMaker JumpStart and Amazon Bedrock. Additionally, SageMaker has released an open-source package called aws/fmeval which can be leveraged to evaluate any type of model for Generation, Summarization, QnA or Classification tasks. In short, FM like Llama2 can be evaluated using,

1. UI based Model Eval: On your Studio UI, navigate to `Jobs` > `Model evaluation`
2. API based Model Eval: Download and use aws/fmeval python API to evaluate LLMs hosted as SageMaker Endpoint

API Based Model Evaluation

> **Important**
> We're going to use `lab-05-llm-eval/Lab_5_FM_Eval_on_SageMaker_Studio.ipynb` notebook for this section

We will evaluate the llama2 7b chat model, which is hosted as a SageMaker endpoint. Alternatively, you have the option to run aws/fmeval on models that are hosted locally, such as on JupyterLab or Code Editor instances, provided they are backed by a GPU.

Our evaluation of the `llama2` 7b chat model will focus on five key metrics:

1. Factual Knowledge
2. Text Summarization
3. Prompt Stereotyping
4. Toxicity
5. Classification

Just like with the other labs, we start off by accepting the EULA

Accept EULA

Please review/accept the Llama2 EULA to proceed. https://ai.meta.com/llama/license/

```
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

```
custom_attribute = f'accept_eula={eula_dropdown.value.lower()}'
print(f"Your Llama2 EULA attribute is set to:", custom_attribute)
```

```
Your Llama2 EULA attribute is set to: accept_eula=true
```

All metric evaluation workflows adhere to a consistent pattern, elaborated in the sub-sections below.

Importing Metric Class and Instantiating a SageMaker Runner

You start off by importing the metric class you plan to evaluate your model on and a built-in optional metric configurator.

example,

```
from fmeval.eval_algorithms.factual_knowledge import FactualKnowledge, FactualKnowledgeConfig
```

OR

```
from fmeval.eval_algorithms.summarization_accuracy import SummarizationAccuracy
```

We can connect to our SageMaker endpoint using a built-in `SageMakerModelRunner` class,

```
sm_fact_model_runner = SageMakerModelRunner(
    endpoint_name=sm_endpoint_name,
    output="[0].generated_text",
    content_template='{"inputs": $prompt , "parameters": {"do_sample": false, "top_p": 0.1, "temperature": 0.1, "max_new_tokens": 128, "
    custom_attributes=custom_attribute,
)
```

For a full list of runners available see here.

Using `SageMakerModelRunner` you can set runtime parameters such as `temperature`, `top_p`, `do_sample`, etc.. and define output format of your model's (ex: `"[0].generated_text"`) response using JMESPath format (https://jmespath.org/).

FMEval Prompt

For a full list of runners available see here.

Using `SageMakerModelRunner` you can set runtime parameters such as `temperature`, `top_p`, `do_sample`, etc.. and define output format of your model's (ex: `"[0].generated_text"`) response using JMESPath format (https://jmespath.org/).

FMEval Prompt

Building on what we learned in Labs 1, 2, and 3 about the significance of effective prompting, we have crafted a straightforward prompt for each test. This is to ensure the model is fully aware of the task it is being asked to perform and understands the expected response. While engineering a prompt for a model's response isn't essential, it does give the model a fair opportunity to succeed in the test. Here is an example of a prompt that we will use for evaluating the model:

```
# prompt for factual knowledge test
prompt_for_fact = """
<s>[INST]
<<SYS>>
Assistant is a expert at fact based question and answers. Assistant must provide an answer to a users question to the best of its knowl

Here are some previous reviews between the Assistant and User:

User: Real Madrid is a soccer club in?
Assistant: Spain

User: Golden Retriever is a breed of
Assistant: Dog

User: Fiji is a country in?
Assistant: Oceania

User: Butter chicken is a curry based dish that originated in
Assistant: Delhi, India

Here is the latest conversation between Assistant and User.

<</SYS>>

$feature

[/INST]
"""
```

OR

```
# prompt for classification task
prompt_for_classification = """
<s>[INST]
<<SYS>>
Assistant is a expert review sentiment text classifier designed to assist respond in only 1's and 0's.

If the provided text has positive sentiment the Assistant responds back with 1. If the provided text has negative sentiment then the As

Here are some previous reviews between the Assistant and User:

User: I have this dress on today in white and i am coming back to buy the second color even though pink is not my favorite. great comfy
Assistant: 1

User: This skirt looks exactly as pictured and fits great. i purchased it a few weeks ago and got lots of compliments on it. however, o
Assistant: 0

User: I purchased the floral patterned version and get complimented every time i wear it. i found it to be pretty true to size, even af
Assistant: 1

User: Fits well through the shoulders and arms, but there is zero waist, and it just looks like a bunch of extra fabric hanging from th
Assistant: 0

User: These run small (i am 110 and got a size 4), they were a tad tight on top. the waist fit but felt a little too snug, short from w
Assistant: 0

User:   Love it! the pants is absolutely beautiful, rich material, it's not your cheap jogger! i am really considering buying a second
Assistant: 1

Here is the latest conversation between Assistant and User.
<</SYS>>

$feature

[/INST]
"""
```

If you wish to evaluate the model on this non-prompted response, just replace

```
prompt = """
....

"""
```

with a simple

```
prompt="$feature"
```

Data Configuration

All the necessary data for conducting tests can be found in the directory `lab-05-llm-eval/sample-datasets`. These sample datasets are provided as examples. In real-world scenarios, you can modify and use your own datasets in this format to conduct custom model evaluations. To begin, you can instantiate a Data Config class as follows:

```
# factual knowledge example
fact_config = DataConfig(
    dataset_name="trex_sample",
    dataset_uri="sample-datasets/trex_sample.jsonl",
    dataset_mime_type=MIME_TYPE_JSONLINES,
```

```
    model_input_location="question",
    target_output_location="answers",
    category_location="knowledge_category",
)
```

OR

```
# classification task example
classif_config = DataConfig(
    dataset_name="classification_sample",
    dataset_uri="sample-datasets/classification_test_clothes.jsonl",
    dataset_mime_type=MIME_TYPE_JSONLINES,
    model_input_location="review_text",
    target_output_location="recommended_ind",
    category_location="category",
)
```

Key components required to instantiate a Data Configurator are data path, input key, output key and data format (ex: `JSON/JSONLINES`).

Running Evaluation

In this last step, we put everything together and run the actual evaluation of LLM. Example shown below,

```
eval_fact_algo = FactualKnowledge(FactualKnowledgeConfig("<OR>"))

# RUN
eval_fact_output = eval_fact_algo.evaluate(
    model=sm_fact_model_runner,
    dataset_config=fact_config,
    prompt_template=prompt_for_fact,
    save=True
)
eval_fact_output = json.loads(json.dumps(eval_fact_output, default=vars))[0]


# print output as dict
print(eval_fact_output)
```

This is repeated for each of the test the LLM is evaluated on with some minor variation with each of the test.

Lab 5 Complete!