# **BUAN 6390.002 - Analytics Practicum**

# **Project FINAL Report**

**GROUP**

Alka Shrestha

Keshav K Rajbux

Navya Bhat

Sai Varshith Komirisetty

Srinaga Mouneeswar Mallipeddi

# Fine TuNNING The gpt-2 Language model

Since its debut in 2019, the GPT-2 Large Language Model, created by OpenAI, has attracted considerable interest. Serving as a cutting-edge natural language processing (NLP) model, it has transformed our interactions with machines and their comprehension of human language. Despite its remarkable capabilities, users frequently desire to fine-tune the GPT-2 model to enhance its suitability for particular tasks and domains.

The GPT-2 model undergoes training on an extensive and varied dataset of diverse text, empowering it to produce coherent and contextually relevant sentences. Nevertheless, its general pre-training may not consistently meet the desired specificity or relevance for certain applications. The process of fine-tuning enables users to adapt the pre-trained GPT-2 model to their specific requirements, thereby enhancing its overall performance and applicability in specialized domains. Fine-tuning GPT-2 becomes essential due to its initial pre-training on a diverse set of text data, which might not precisely match the requirements. To tailor GPT-2 for crafting content in the designated area, like generating recipes, fine-tuning is employed. This method entails adjusting GPT-2's weights using a more focused dataset, custom-tailored the requests. This ensures a more adept grasp of the nuances, vocabulary, and style pertinent for particular tasks and domains.

**Benefits of Fine Tuning:**

Fine-tuning offers numerous advantages, and among them are a few noteworthy ones:

1. **Improved Domain Specificity** - The process of fine-tuning the GPT-2 model with domain-specific data guarantees the generation of outputs that are not only more accurate but also highly relevant to the specific industry or field of interest.
2. **Enhanced Model Performance** - Through training on a targeted dataset, fine-tuning mitigates errors and enhances the model's comprehension of specialized jargon or terminology, resulting in outputs of superior quality.
3. **Increased Efficiency** - A GPT-2 model that has undergone fine-tuning exhibits increased efficiency in task completion, thanks to its alignment with the specific requirements of the given application.

It's important to note that while fine-tuning offers these benefits, it also requires careful consideration, as improper fine-tuning may lead to overfitting or unintended consequences. Therefore, best practices and a thoughtful approach are recommended when fine-tuning GPT-2 or any other language model.

**How to Fine-tune GPT-2:**

Fine-tuning GPT-2 incorporates a range of methods and tools, influenced by factors such as data, resources, and objectives. The process begins by choosing a GPT-2 model size, which can vary from small to extra-large. Following this, you must curate and preprocess a dataset customized to your specific domain or task.

The utilization of frameworks and libraries like Hugging Face Transformers or PyTorch is integral to the fine-tuning of GPT-2. This involves adjusting hyperparameters such as the learning rate, batch size, number of epochs, and optimizer. Concurrently, it is essential to define a loss function that measures the difference between the model's output and the expected outcome.

To evaluate and validate your model, various metrics such as perplexity, accuracy, or human evaluation can be applied. Ultimately, generating samples from the test set facilitates a comparison with the original data, enabling the assessment of factors such as quality, diversity, coherence, relevance of the generated text, and the identification of any errors or biases.

**Best Practices for Fine-Tuning GPT-2**

* Selecting an Appropriate Dataset
* Data Preprocessing
* Hyperparameter Tuning
* Regularization Techniques
* Evaluation and Monitoring

# Metrics USed

1. **Burstiness**

In the realm of written content, incorporating burstiness introduces an additional layer of excitement and captivation. This technique strategically integrates bursts of information and engaging elements into the text, aiming to keep readers enthralled and eagerly anticipating more.

Consider burstiness in writing as akin to a roller coaster ride, where the content takes readers on thrilling twists and turns. Achieving the desired effect involves maintaining a balance in rhythm and flow, alternating between longer, intricate sentences and shorter, punchy ones. This intentional variation results in a dynamic reading experience.

Picture embarking on a literary adventure where each paragraph surprises you with unexpected facts, intriguing anecdotes, or thought-provoking insights. Longer sentences paint vivid landscapes, setting the stage for an immersive experience. Suddenly, a short sentence emerges, delivering a sharp and concise impact, reminiscent of a burst of fireworks.

To achieve a heightened burstiness effect, I'll employ a thoughtful blend of sentence structures, varying lengths, and a sprinkle of rhetorical devices. This combination of writing techniques will infuse energy and momentum into the content, ensuring readers remain engaged and eagerly absorb each word.

It's crucial to note, however, that burstiness should seamlessly complement the overall purpose and logical flow of the text. Think of it as seasoning in a gourmet dish—just enough to enhance the flavor without overpowering the main ingredients. This way, the content maintains its professional, informative, and objective tone while adding that extra spark that keeps readers on the edge of their seats.

1. **Perplexity**

Perplexity functions as a crucial metric in assessing the effectiveness of language models, gauging their ability to predict the subsequent word in a sequence. As AI-generated text unfolds procedurally, selecting each word based on probabilities derived from a sample, perplexity becomes instrumental in understanding the model's predictive performance.

In the context of writing, perplexity serves as a yardstick for evaluating text quality and coherence by scrutinizing the model's aptitude in predicting upcoming words within a given context. The measured perplexity not only reflects the fluency of the text but also provides valuable insights into the model's capacity to accurately capture the intended meaning. A lower perplexity value indicates superior predictive abilities and a heightened level of comprehension for the reader.

Proficient human writers typically produce content characterized by low perplexity. Their expertise manifests in the meticulous selection of words, the purposeful construction of sentences, and the seamless connection of ideas, resulting in a coherent and easily understandable text.

In contrast, AI-generated content often displays higher perplexity. Despite grammatical correctness, language models may lack human-like coherence and a nuanced understanding of context. This disparity can lead to predictions that, while grammatically sound, deviate from the intended meaning, thereby elevating perplexity.

The assessment of perplexity emerges as a valuable tool in discerning between AI and human writing, emphasizing the critical aspects of coherence and appropriateness. It assumes a pivotal role in the evaluation and comparison of content quality.

# FINE Tuning Method USed

Fine-tuning is a crucial step in enhancing the performance of pre-trained models, and the PyTorch Lightning API provides a streamlined and efficient framework for carrying out this process. Let's delve into an introduction to fine-tuning using the PyTorch Lightning API.

PyTorch Lightning simplifies the fine-tuning process by offering a high-level interface that abstracts away many of the complexities involved. The following steps provide a general overview of how fine-tuning with PyTorch Lightning is usually conducted:

1. Import Necessary Libraries: Begin by importing the essential libraries, including PyTorch and PyTorch Lightning. Ensure that you have the required dependencies installed.
2. Define the Dataset: Prepare your dataset, ensuring it is compatible with the pre-trained model you intend to fine-tune. PyTorch provides various datasets and transformations for common use cases.
3. Define the LightningModule: Create a LightningModule that encapsulates your pre-trained model and the fine-tuning logic. This involves modifying the model architecture and specifying the training and validation steps.
4. Initialize PyTorch Lightning Trainer: Set up the PyTorch Lightning Trainer, specifying essential configurations such as the number of training epochs, batch size, and any other relevant parameters.
5. Start Fine-Tuning: Initiate the fine-tuning process by calling the `fit` method on the Trainer, providing the LightningModule and DataLoader.

### References