**WikiMind: Training a Minimal-Sized Language Model on Wikipedia**

This project fine-tunes a minimal GPT-2 model using a curated mix of Wikipedia articles and generated Q&A pairs. The model uses efficient tokenization and a resource-conscious training strategy to generate coherent, structured responses. An interactive command-line Interface demonstrates its ability to handle both descriptive and dialog-style inputs. The pipeline is scalable and ideal for building lightweight educational tools, chatbots, or domain-specific assistants.

**Justification(s)**

**Architecture Justification: GPT-2 Small**

* Training from scratch was ruled out due to lack of computing resources, as I didn’t have the computational resources to train a model from scratch.
* GPT-2 Small is a decoder-only transformer with ~124 million parameters. It follows the causal (autoregressive) language modeling objective, predicting the next token given all previous tokens. Built with 12 transformer blocks, 12 self-attention heads, and a hidden size of 768.
* **Decoder-Only Transformer:** GPT-2 small uses a unidirectional transformer (decoder-only), making it ideal for tasks like text generation, auto-completion, and Q&A generation where future context isn't known.
* **Minimal Yet Capable:** The small variant of GPT-2 (~124 million parameters) strikes a strong balance between Computational efficiency and generation quality, hence producing fluent, diverse, and coherent text with proper fine-tuning.
* GPT-2 small was trained on a diverse and massive corpus (WebText), giving it a general understanding of language.
* **Layer Normalization & Activation Stability:** GPT-2 uses **pre-layer normalization**, which stabilizes training and reduces exploding/vanishing gradients in deep transformers — useful when fine-tuning on relatively smaller datasets. It works well with **bf16 or mixed-precision training**, reducing memory use and training time.
* GPT-2 handles sequences up to 1024 tokens, which is more than sufficient for Wikipedia-derived Q&A (avg. ≤ 200 tokens).
* Since it's autoregressive and Q&A follows sequential logic, it naturally models dependencies between the question-answer pairs and articles.

**Alternate Considered:**

**T5-Small (Encoder-Decoder Model):**

* **Why Considered:** 
  + T5 is a versatile text-to-text transformer that can perform tasks like summarization, translation, and Q&A by converting everything into a text-in/text-out format.
  + The T5-Small version (~60M parameters) is compact and efficient enough to run on mid-sized hardware.
* **Why not chosen:** 
  + T5 uses an encoder-decoder architecture, which is better suited for tasks where full context is available upfront (e.g., translation, summarization).
  + For left-to-right text generation like free-form Q&A, decoder-only models like GPT-2 are more natural and efficient, as they are designed to predict the next token sequentially.
  + GPT-2 also supports faster generation during inference since it doesn’t require encoding the full input every time.
  + T5's sequence-to-sequence format requires reformatting input/output pipelines, which adds overhead to a lightweight prototype like this.

**Tokenizer Justification: GPT-2 Tokenizer (Byte-Pair Encoding)**

* Used GPT2Tokenizer from Hugging Face, which employs Byte-Pair Encoding (BPE). The GPT-2 ‘s BPE replicates the exact vocabulary and tokenization process used during GPT-2's pre-training.
* Perfectly aligned with the architecture of GPT-2, ensuring token IDs match the pre-trained embeddings;

Mismatching tokenizers can lead to invalid input embeddings; using the original tokenizer ensures compatibility.

* BPE provides subword-level tokenization: BPE splits rare or unseen words into subword units for better generalization.
  + Efficiently handles rare words, named entities, and technical terms which is especially important for Wikipedia where many rare, domain-specific terms exist.
  + Prevents vocabulary explosion while maintaining good coverage as subword models avoid needing huge vocabularies while still covering diverse language.
* Robust to diverse Wikipedia text, this tokenizer was trained on web text and is highly resilient to varied token structures. Including the following:
  + Multilingual content
  + Punctuation, math symbols, URLs, and special characters
* Avoids out-of-vocabulary (OOV) issues that would arise with word-level tokenizers as BPE ensures everything is tokenizable into known subunits.
* GPT-2 does not have a built-in <pad> token, so <eos> was reused as a padding token — a standard and compatible workaround.
* Hugging Face's DataCollatorForLanguageModeling seamlessly supports this setup during batch training especially when using mlm=False, it handles padding and masking automatically.
* BPE tokenization results in faster training and inference, with lower memory overhead compared to character-level models. Accurate – character-level tokenization significantly increases sequence length, slowing down transformer models.

**Alternate Considered:** SentencePiece / WordPiece

* SentencePiece or WordPiece were not used because they are designed for encoder or encoder-decoder architectures like BERT or T5, and do not match GPT-2’s pre-trained tokenization scheme.
* It was avoided because they are incompatible with GPT-2’s autoregressive, decoder-only setup. It could be used be used for other models such as BERT , T5 etc.

**Training Strategy Justification – Fine-Tuning GPT-2 on Limited Wikipedia Q&A Data**

***Why Fine-Tuning GPT-2 small?***

* Fine-tuning was selected instead of training from scratch to utilize GPT-2’s pre-existing language capabilities, minimize computing demands, and speed up convergence. Given my limited computational resources, training the model from the scratch wasn't feasible. Although fine-tuning could be enhanced by increasing the number of articles, my computational constraints forced me to keep it limited. However, I conducted thorough testing beforehand submission.
* **Fine-Tuning vs Training from Scratch**
  + **Justification:** Fine-tuning a pre-trained model like GPT-2 allows us to leverage its extensive linguistic and semantic understanding gained from large-scale training on general internet data.
  + This significantly reduces computational cost, training time, and data requirements while still achieving high-quality task-specific adaptation which is important as I’m working with a setup that has limited Computational resources.
* **Data Design and Preparation for fine tuning:**
  + **Justification:**
    - The text was cleaned and formatted to remove noise and ensure input consistency.
    - Q&A pairs were generated using mosaicml/mpt-7b-instruct to automate structured annotation at scale, reducing manual labor while maintaining output quality. This could be done more efficiently with OpenAI API , but it is not a cost-effective solution.
    - Combining narrative article text with Q&A data helps the model learn both factual writing and interactive question answering, enhancing its flexibility.
    - Shuffling the dataset avoids learning order bias and encourages generalization.
* **Tokenization and Dataset Splitting**
  + **Justification**: GPT2Tokenizer was used to **preserve vocabulary consistency** with the pre-trained model.
  + Tokenizing both article and Q&A data ensures uniformity in representation, and the data was split into **training (90%) and validation (10%)** to enable effective performance tracking.
* **Training Configuration:**
  + **Justification**: The configuration was optimized for efficient training under compute constraints:
    - batch\_size = 8: Suited for A100 GPU memory.
    - num\_train\_epochs = 5: Provides sufficient exposure to the limited dataset without overfitting.
    - learning\_rate = 3e-4: Empirically stable for transformer fine-tuning.
    - bf16 = True: Reduces memory usage and speeds up training with minimal precision loss.
    - gradient\_accumulation\_steps = 4: Simulates a larger batch size for more stable updates.
  + Hugging Face's Trainer API was used to simplify training orchestration, logging, checkpointing, and evaluation.
* **Evaluation Strategy**
  + **Justification**: The model was evaluated using **epoch-wise validation loss and training loss**, allowing us to monitor overfitting and convergence trends.
  + **Manual inspection** of the generated Q&A outputs ensured that the model was producing **contextually relevant and syntactically correct responses**.The integration of automated evaluation metrics, such as ROUGE, for the quantitative assessment of output quality was contemplated; however, due to constraints in computational power, it could not be implemented
* **Resource-Conscious Design**
  + **Justification**: All design choices — from data size to model size — were model considering limited computational resources, with the aim to create a **functional, demonstrable model** that could be scaled in the future with larger datasets and longer training.
  + While the current setup delivers meaningful results, there were several enhancements I originally planned but was unable to implement due to resource constraints:
    - Training on a **larger, more diverse dataset** for improved generalization.
    - Integrating **automated evaluation metrics** like **ROUGE** to quantitatively assess output quality.
    - Experimenting with longer sequences and additional tuning.

Despite these limitations, the model successfully learned to generate coherent and structured answers. The pipeline remains **scalable** for future improvements when more computing becomes available.

* **Inference Setup** 
  + After fine-tuning, the model was integrated into an interactive command-line chatbot to evaluate its ability to generate coherent and structured responses to user questions.
  + **Justification and Key Features:**
    - **Purpose**: This function provides a simple CLI-based interface for testing the fine-tuned model's ability to answer custom questions in real time.
    - **Q&A Prompting Format**: The input follows the same "Q: ... A:" structure used during training, allowing the model to continue the pattern naturally.
    - **Generation Parameters**: top\_p = 0.9 and temperature = 0.7 enable **controlled diversity**. repetition\_penalty = 1.2 avoids repeating words or phrases.early\_stopping = True ensures more **concise, readable outputs**.
    - **Scalable Setup**: This script is lightweight and can easily be extended to a web-based UI or API for production or demo use.