Model:
```
# Step 4: Model loading with enhanced error handling
MODEL_NAME = "microsoft/phi-1_5"
```
Use phi 1_5 instead of 1 or 2.

Dataset:
Use plain text TXT files, including manual checks and uploads.

Notebook Editor:
Use Kaggle instead of Colab due to the RAM limit of each session for "free accounts".

When running a session: turn on the internet option (but has a limit per week)

Estimated Time Span:
For training the dataset of plain text consisting of a 3 page long, font size 11 google doc, it would generally take about 20 minutes for only the training part and the whole running might take 30-40 mins.

Total Time Running the Entire Notebook: 30 mins.

Model not generating texts. (have to run from the first cell).
"Trans_0516_kag_DB2" is using the modified, more comprehensive prompt answering more questions, while "trans-0516-db1" is using shorter but clearer contexts surrounding only a few questions. (* Using papers are better due to their inherent clarity of words)

- **How to resolve the time waste and low efficiency of pre-training?**
- **How to enhance the contextual accuracy (lang chain) in the answers.**
  - **Prompt Engineering.**
    - Empty prompt is giving weirdly intangible information, likely embedded in the default answer of the model.
    - Define certain styles of answering by asking with key words like "explain … in simple terms to the public audience. Answer: …" (and followed by answers to clarify the styles). Also, for "Explain … Answer: …"
    - There should not be visible python outputs while explaining the prompts.
    - Define whether or not something is the proper short form of specific terms. Define the uniqueness of terminology.
    - Limit the domain to answer only tangible questions and terms/concepts. Nicely rejecting to give out answers to questions about terms/concepts out of the domain specific to crypto wallet.
    - There should not be questions I did not ask. (limit the extent of hallucination and incorrect form of interconnected thinking).
    - Answers should align with the question.

- **Pro Tips:** Use scientific journals, papers instead of unverified web crawl. Remove misinformation, and human review for critical domains. Ensure broad domain representation to prevent overfitting to niche or biased sources. Use entity linking. Jointly train on language modeling to predict the next token and refine the prompt (optionally), be able to verify facts, answers with evidence alike Natural Questions. Apply contrastive learning to penalize the model for generating outputs that contradict verified facts such as using negative samples of incorrect statements. Inject adversarial, counterfactual, and ambiguous examples during this pre-training stage to teach the model to recognize and avoid hallucinations. Prioritize data quality over quantity (as showcased by the 2 versions of pre-training with same structure but different database contents). Train the model to condition its outputs on retrieved documents. Incorporate human-annotated flags by using datasets where hallucinations are explicitly labeled.

How to use algorithms to identify, filter/classify specific domains and only answer domain-specific questions.

Clean up or comment out the lines used for checking cache.

Additional High-Quality Domain-Specific Sources to Consider: Whitepapers about Bitcoin, Ethereum, MetaMask docs, Technical blog posts alike Coinbase, Binance Academy, Crypto wallet FAQs & user guides from Trezor.

Format: Structured text with clear headings like "How Crypto Wallets Work", "Public VS Private Keys", "Hot Wallets VS Cold Wallets", "Multi-Signature Wallets."

Then, Start Fine-Tuning: Using Supervised Learning on a labeled dataset. Curate a Q&A-style dataset, such as "What is a seed phrase?" → "A seed phrase is a 12-24 word backup…" And, use LoRA (the Low-Rank Adaptation) for efficient fine-tuning if the computing ability is limited. Use GPT-4 or Claude to generate 50+ synthetic Q&A pairs about crypto wallets. Then fine-tune on this synthetic dataset. Lastly, evaluate the model on crypto wallet-specific questions, refine by adding missing topics, use few-shot prompting to guide the model during inference.