Pico-LLM Core Task Action Plan

Team: ________________________

November 11, 2025

Current State Assessment

- The training loop, dataset mixer, and generation harness are in place. The default configuration trains only the LSTM model on TinyStories.
- `KGramMLPSeqModel` lacks an initialized network; the forward pass currently constructs one-hot contexts but forwards them through `self.net = None`.
- `nucleus_sampling` performs greedy decoding (`argmax`) instead of top-p sampling.
- `RMSNorm` and `TransformerModel` are unimplemented stubs.
- The main script does not yet instantiate or train the k-gram MLP or Transformer models, so only the LSTM contributes to outputs and figures.

Implementation Plan (Core Tasks Only)

1. **Sanity-check the starter pipeline.**
   - Run the existing configuration with reduced hyperparameters (e.g., `--block_size 32`, smaller batch size) to confirm end-to-end execution on TinyStories or on the provided `3seqs.txt` fallback.
   - Verify device selection logic and document any environment-specific tweaks needed for CPU-only runs.
2. **Complete `KGramMLPSeqModel`.**
   - Design `self.net` as an MLP that consumes the flattened one-hot (or embedded) k-gram context and outputs vocabulary-sized logits.
   - Consider replacing explicit one-hot flattening with an embedding-based projection for efficiency if profiling reveals bottlenecks.
   - Validate training by comparing loss curves and sample generations against the LSTM baseline.
3. **Implement true nucleus (top-p) sampling.**
   - Convert logits to probabilities via softmax, sort tokens by probability mass, accumulate until the cumulative sum first exceeds the user-specified p, then normalize and sample within that subset using `torch.multinomial`.

- Exercise the sampler with multiple p values (e.g., 0.8, 0.95, 1.0) to ensure qualitative differences show up in generated stories.

4. **Build the Transformer stack.**
   - Implement `RMSNorm` with learnable weight, computing $\mathrm{RMS}(x)$ per token and returning $x/\mathrm{RMS}(x) \times w$.
   - Construct `TransformerModel` with:
     4.a. Token embeddings and a positional encoding scheme (e.g., learned positional embeddings) compatible with the existing `block_size`.
     4.b. A configurable number of decoder blocks, each containing multi-head causal self-attention with residual connections, followed by an MLP (e.g., two-layer feed-forward with SiLU) and a trailing `RMSNorm`.
     4.c. Rotary or additive causal masking to prevent attention from peeking at future tokens.
     4.d. A final projection layer that maps hidden states back to the vocabulary dimension for next-token prediction.
   - Integrate the Transformer into `main()` by adding it to the `models` dictionary (optionally re-enabling the k-gram model once complete).

5. **Unify training and evaluation across models.**
   - Confirm that all models train under the same dataloader and loss function without numerical instability, adjusting learning-rate or gradient clipping if needed.
   - Capture at least three figures required for the oral presentation (e.g., training loss trajectories per model, qualitative sample comparisons, attention-head visualizations for the Transformer when feasible).
   - Log configuration details, encountered issues, and fixes to streamline the final presentation narrative.

6. **Prepare for the oral evaluation.**
   - Schedule the presentation slot, list all collaborators, and rehearse a shared walkthrough that covers implementation decisions, debugging stories, and sample outputs.
   - Assemble sources consulted (LLMs, forums, papers) to reference during Q&A, matching the course policy.

Risk Mitigation and Tracking

- Start with toy datasets and short sequences when bringing new models online to avoid long feedback cycles.
- Check gradients and activations for exploding/vanishing behavior as each component is introduced, especially within the Transformer block.
- Maintain incremental checkpoints (even for small models) so that generated samples and loss curves can be reproduced quickly for figure generation.