Architecture Document

Email-Drafter Agent (GPT-2 Medium + LoRA)

Daksh Yadav

September 15, 2025

1 Overview

The Email-Drafter Agent is a lightweight system that fine-tunes a GPT-2 Medium base model with LoRA adapters to generate polite, structured emails (e.g., assignment extensions, recommendations, leave requests). The system accepts an instruction-style prompt and returns a polished email draft, exposed via a Gradio web UI with deterministic post-processing to enforce structure.

2 High-Level Components

1. **Dataset:** JSONL files with {`"prompt"`, `"completion"`} pairs. Example sizes: 2000 train, 200 validation.
2. **Tokenizer:** GPT-2 BPE tokenizer with an added `[PAD]` token.
3. **Base model:** `gpt2-medium` (causal LM).
4. **PEFT adapter:** LoRA on attention projection matrices (`c_attn`); only LoRA weights are trainable.
5. **Training:** Hugging Face `Trainer` with gradient accumulation to simulate larger batch sizes on 8GB GPUs.
6. **Post-processing:** `cleanup_and_validate` enforces subject, greeting, closing, and fixes year normalization.
7. **UI:** Gradio front-end for input, generation, editing, and download.
8. **Logging & eval:** Training/eval loss, pass-rate on validation, and interaction logs (CSV).

3 Design Rationale

- **GPT-2 Medium:** balance between quality and ability to train on a single 8GB GPU.
- **LoRA:** parameter-efficient fine-tuning, producing small adapter files instead of full checkpoints.
- **Post-processing:** ensures structural reliability for downstream users.
- **Gradio UI:** lightweight, interactive demo suitable for deployment and grading.

4 Data and Control Flow

1. User submits instruction (prompt) in UI.
2. Tokenizer encodes the prompt $\rightarrow$ input IDs.
3. Base model + LoRA adapter generate raw text.
4. Post-processor validates and enforces subject, greeting, closing, and year normalization.
5. If validation fails, a fallback template is returned.
6. Final draft displayed in UI for edits/download.

5 Architecture Diagram

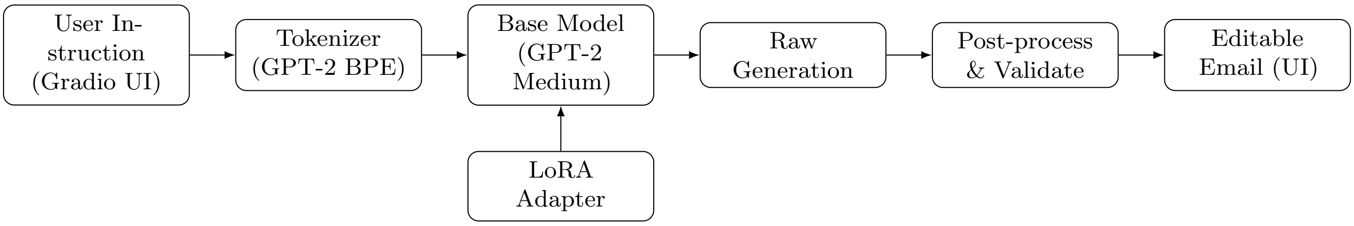

Figure 1: System architecture (inference path).

6 Training Setup and Hyperparameters

Parameter	Value
Base model	`gpt2-medium`
PEFT method	LoRA
LoRA config	$r = 8$, $\alpha = 32$, dropout = 0.1
Train / Validation size	2000 / 200
Max sequence length	384
Batch size	1 (grad. accum. 8 → effective 8)
Epochs	8
Learning rate	2×10^{-4}
Precision	FP16
Save	LoRA adapter in `lora-output/`

Table 1: Training hyperparameters

Training Results

- Final training loss: 0.0751
- Final validation loss: 0.0693
- Perplexity $\approx e^{0.0693} = 1.07$

7 Interaction Flow (Detailed)

Figure-1 already illustrates the inference pipeline. Below we expand the flow into numbered stages:

1. **User Instruction:** User types a natural-language request (e.g., "Draft an email to Prof. Singh requesting a 3-day extension") into the Gradio interface.
2. **Preprocessing:** Prompt is minimally cleaned (e.g., strip whitespace, standardize quotes).
3. **Tokenization:** Hugging Face `AutoTokenizer` converts text into subword tokens with attention masks. A `[PAD]` token ensures consistent batch lengths.
4. **Model Inference:** The base GPT-2 Medium model, augmented with LoRA adapters, produces a sequence of token IDs using sampling or beam search.
5. **Post-processing:**
   - Check structural validity (subject line, greeting, closing).
   - Apply heuristics (normalize year to 2025, enforce polite closing).
   - If invalid → retry with adjusted decoding or return a safe template.

6. **User Delivery:** The polished email draft is shown in the UI for optional editing and download.
7. **Logging:** Prompt, raw generation, final cleaned output, and validity flag are appended to a CSV/JSON log for evaluation.

8 Models Used

- **Base model:** `gpt2-medium`, 345M parameters.
  - Widely supported in Hugging Face Transformers.
  - Fits within an 8GB GPU when fine-tuned with parameter-efficient methods.
  - Adequate for sentence- and paragraph-length email drafts.
- **Tokenizer:** GPT-2 BPE with added `[PAD]` token for batching.
- **PEFT Adapter:** LoRA with $r = 8$, $\alpha = 32$, dropout = 0.1, attached to attention projections.

9 Reasons for Design Choices

GPT-2 Medium A balance of fluency and resource efficiency. Larger models (GPT-2 XL, GPT-3) provide better coherence but require hardware not available in the target deployment (Colab / local GPU).

LoRA Reduces trainable parameters by $> 95\%$, making training feasible on commodity GPUs while keeping quality close to full fine-tuning.

Gradio Lightweight, Python-native interface for rapid prototyping. Avoids building a full web stack while still enabling demos and user testing.

Deterministic Post-processing Guarantees minimum structural quality. This mitigates common LM failure modes (missing subject lines, wrong years).

10 Deployment and Scalability Notes

- Current demo runs in Google Colab with a Gradio UI.
- For production, replace Gradio with a REST API (FastAPI/Flask) serving the LoRA-adapted model.
- Containerize with Docker for reproducibility.
- Scale inference with batching or quantization (e.g., 8-bit weights).
- LoRA adapter keeps storage light (a few MB) vs full model checkpoints (hundreds of MB).

11 Limitations

- Training data limited to polite academic-style emails (risk of narrow generalization).
- Base model occasionally produces verbose or repetitive outputs.
- Post-processing covers common structural errors but not deep semantic accuracy.