Smart Lead Gen: A Generative AI for Hyper-Personalized Professional Outreach

- Milestone 3: Model Training.
- Team: Group 7
- Course: DS and AI Lab
- Submission date: November 4th, 2025

Stage A: Personality classification Module
Stage B: Inference Pipeline
Stage C: Email Generation Module

The primary objective of Stage 2 is to provide deep psychographic enrichment **to** our outreach pipeline. This module is designed to infer a target's 16-type MBTI personality profile by analyzing their publicly available written text. This inferred type serves as a critical input for Stage 3, allowing the generative model to precisely tailor the outreach message's tone and style to resonate with the recipient.

Phase A

This one-time process created our specialized classifier. We used transfer learning to adapt a pre-trained model for our specific task.

Model Architecture & Parameters

- Base Model: We selected distilbert-base-uncased, a powerful Transformer model chosen for its high performance and efficiency.
- Parameters: This model has approximately 66 million parameters.
- Weight Initialization: We loaded the 66 million pre-trained parameters for language understanding. We then attached a new, uninitialized classification head (the pre_classifier and classifier layers) to predict our 16 labels.
- Our fine-tuning process trained these new layers from scratch while fine-tuning the 66 million base parameters.

Training Dataset & Preprocessing

- **Data Source:** We used a merged corpus of the Myers-Briggs and Pandora (Big 5) datasets, loaded from preprocessed_personality_data.csv.
- **Data Sampling:** From a total of 2,332,229 available rows, we took a random sample of 100,000 rows for efficient training.
- **Text Cleaning:** Our clean_text_for_training function was applied to all samples, which involved:
    - Converting text to lowercase.
    - Removing all URLs (http, https, www).
    - Removing all @mentions.
- **Tokenization:** We used the distilbert-base-uncased tokenizer to convert the cleaned text into numerical IDs. We set a uniform sequence length by padding and truncating all samples to a max_length of 256 tokens.

Training & Checkpointing

- **Training:** The model was fine-tuned on a GPU (cuda) for 3 epochs with a TRAIN_BATCH_SIZE of 16.
- **Model Checkpoint:** After training, we saved the entire fine-tuned model (all 66M+ updated parameters) and its tokenizer to the ./my_mbti_classifier directory. This checkpoint now contains all our specialized knowledge.

Phase B

This is the process that runs live in our application, using the checkpoint created in Phase A.

Loading the Classifier

- The PersonalityClassifier class is initialized by loading our saved model checkpoint from ./my_mbti_classifier. It does not load the generic base model; it loads our already-trained version.
- The model's config.json file, saved during checkpointing, contains the crucial id2label and label2id mappings (e.g., {8: "ISFJ"}). This allows the model to output human-readable labels.

Live Data Processing

- **Input:** The pipeline receives the structured JSON from Stage 1.
- **Text Aggregation:** The process_stage_1_output function extracts and combines text from multiple fields (headline, career_history.description, skills, recent_activity, etc.) into a single string for analysis.
- **Inference:** This combined text is cleaned, tokenized, and fed to our loaded model.
- **Prediction:** The model outputs a numerical ID (a logit), which is converted to the highest-probability class ID (e.g., 8).
- **Translation:** Using the saved id2label map, the model translates this ID into the final string label (e.g., "ISFJ").

Output

- The final predicted string (e.g., "ISFJ") is added to the JSON object under the new key "inferred_mbti".
- This "enriched" JSON is then passed to Stage C.

Phase C

1. Model Exploration and Evolution

The development started with Mistral-7B-Instruct-v0.2, an instruction-tuned model used for initial testing. Although it produced good results, it required significant compute time and resources, making it less suitable for fast iterations.

To address this, we transitioned to TinyLlama-1.1B-Chat-v1.0, a lightweight, efficient model that loads quickly in 4-bit quantization and performs reliably even on modest hardware. Its responses are concise, coherent, and well-structured, making it ideal for short email generation.

We later experimented with LLAMA-3-8B enhanced with LoRA (Low-Rank Adaptation) for tone fine-tuning and improved personalization. However, during inference, we encountered some issues. These are currently being debugged. Meanwhile, TinyLlama-1.1B has been chosen as the operational model for ongoing use.

2. Functional System Design

The email generation framework integrates multiple components:

- **Chat Framework:** Implements a unified chat() function using Hugging Face's transformers library, handling prompt creation, message formatting, and controlled generation.
- **Profile Normalization:** Functions like normalize_profile() and build_lead_profile() ensure consistency by cleaning and structuring lead data.
- **MBTI-Based Tone Modulation:** Each MBTI type (e.g., ENTJ, INTJ, ENFP) maps to parameters such as assertiveness, formality, and CTA style, forming a dynamic style profile.
- **Prompt Engineering:** A clear system prompt enforces **structure: 60-120 words**, one call to action, no external links, and inclusion of greeting and closing.
- **Output Parsing and Post-Processing:** Extracts and validates JSON output, adds fallback handling, and appends a short compliance footer.

3. Testing and Validation

The model was tested using dummy data with **JSON-style input**. The system successfully generated short, context-aware, polite cold emails that followed the required structure and tone guidelines.