**CS4100 Project: Anime Character Simulator**

Chia-Yi Hou

**Description of the Project:**

As an anime fan, I always wish to have a chance to talk to the characters and get replies from them, as if we lived in the same world. Although some Otome games (dating-simulating games targeted at females) allow players to select replies, those dialogue options are still pre-set and lack flexibility, so I often feel none of the choices fully express what I really want to say. That is why I have always hoped for a system where I can type freely and get responses that reflect the character’s personality. This idea became the inspiration for my current project, which is building a character chatbot.

There are similar models on the market, and even ChatGPT can do similar things for some characters. Those models usually allow users to specify characteristics and use different models according to the instructions. However, most of the responses are either out of character or too conservative to reflect the characters’ more complex or varied personalities, such that a calm character would never be funny in any way. As recent anime characters have become more realistic and have more different performances in different contexts, capturing this kind of variability has become more important.

In class, we are taught that models “learn” from the pattern of our data, and bias in data could also make the model biased. For example, if all the training text is collected from Reddit, then the model after training might sound more radical and informal or even generate misleading or factually incorrect content, depending on the training source. This inspired me to take advantage of this kind of bias so the model can not only talk like the characters but also capture aspects of their thinking style, since the adapter is trained exclusively on their dialogue. In this project, I am going to use the LoRA technique to fine-tune the already well-trained Japanese GPT model, as if adding a character-specific layer that lets the model 'pretend' to be that character.

**What is the Data?**

In this project, I used three different datasets for different stages of training, which are general dialogue data from the anime, *Blue Lock*, character-specific lines from my target character Hyoma Chigiri, and a conversational dataset with tones labeled called J-CHAT.

**Dataset 1: Blue Lock Dialogue Dataset:**

The general Blue Lock dialogue dataset is compiled from the subtitles of the first season of the anime *Blue Lock*, which contains 24 episodes. The subtitle files were downloaded from [Kitsunekko.net](https://kitsunekko.net/dirlist.php?dir=subtitles%2Fjapanese%2FBlue+Lock%2F), a well-known site for Japanese subtitle collections. Based on the file names, these subtitles appear to be Netfilx versions.

The subtitle files are already sorted chronologically using timestamp information. While I acknowledge that not every timestamped line perfectly corresponds to a single character’s utterance, I treat each line as an individual dialogue turn, as such imperfections are unlikely to affect the model’s learning, since the goal is not precise speaker tracking but stylistic adaptation. This assumption, though imperfect, provides a simple and consistent way to split and structure the data for training.

I pair each line with one immediately preceding it, creating previous line and current line pairs (column names being “prev\_line” and “current\_line”). These pairs simulate conversational flow and are used to train the model to understand basic context and turn-taking patterns. This dataset would be used in the first stage of fine-tuning a Japanese GPT model. The goal is to expose the model to the linguistic style, tone, and worldbuilding specific to *Blue Lock*.

**Dataset 2: Chigiri-specific Dataset:**

The Chigiri-specific dataset contains only the lines spoken by Hyoma Chigiri and the previous lines. These lines were manually extracted from dataset 1, the full Blue Lock dataset, with the goal of creating a cleaner, more accurate subset for character-level fine-tuning. For lines that are first in a conversation or appear to be an internal monologue, I add system-level indicators such as “独り言” (soliloquy) or other placeholders used in Rinna’s prompting format.

Since Chigiri is the target character of this project and only has around 270 lines in total, it was important to ensure high-quality data, even if it meant manually refining and pairing the dialogue lines.

After formatting the Chigiri lines into previous-current pairs as Dataset 1, I used a BERT-based embedding to vectorize each line and applied K-Means clustering (k=10) to group them. These clusters were then used to assign initial tone labels, which I further adjusted manually. This tone-labeled dataset is later used to train a tone prediction model using Feed-Forward Neural Network (FFNN). This dataset would also be used in the second stage of fine-tuning the Japanese GPT model.

**Dataset 3: J-CHAT:**

This dataset is extracted from [EaST-MELD](https://github.com/ku-nlp/EaST-MELD?tab=readme-ov-file), the Japanese-translated version of MELD, which is a dialogue emotion classification dataset based on the TV show *Friends*.

From EaST-MELD, I selected three columns: the Japanese-translated dialogues (Text(Ja)), the emotion label (Emotion), and the dialogue grouping (dialogue\_id). Following the same approach as with Dataset 1, the Blue Lock dataset, I paired each line with its immediate predecessor to form previous-current line pairs. Two specific adjustments were made to the pairing process for the J-CHAT dataset, compared to Dataset 1. If a line or its context was missing a Japanese translation, the pair was skipped to avoid data inconsistency. Additionally, pairs were only formed within the same dialogue\_id to ensure that each interaction remained within its original conversational context. The final dataset contains columns “prev\_line”, “current\_line”, and “tone”, and is used to train the initial version of the tone prediction FFNN model.

While the emotional tone in American sitcoms may differ from how it is in anime, this dataset provides a valuable starting point for training the FFNN model, which is built from scratch and benefits from a large and labeled dataset during training.

**How do Models Work?**

In this project, I used two pre-trained models, Rinna’s GPT model and cl-tohoku/bert-base-japanese, as well as a self-built Feed-Forward neural Network (FFNN).

The BERT-based model was mainly used to convert Japanese sentences into fixed-size embeddings, which are then used for clustering or tone classification. The FFNN is trained to predict the tone that a character would use to respond to a given user input. This prediction is then used to select an appropriate pre-written prompt that guides the GPT model’s response generation.

Initially, I planned to use Rinna's [japanese-gpt-neox-3.6b-instruction-sft](https://www.aimodels.fyi/models/huggingFace/japanese-gpt-neox-36b-instruction-sft-rinna), which is instruction-tuned and supports structured prompting (e.g., role, user input, response). However, due to hardware limitations, I switched to the smaller [japanese-gpt2-medium](https://www.aimodels.fyi/models/huggingFace/japanese-gpt2-medium-rinna), which does not support instruction tuning out of the box. Still, I manually included the prompt format in my training data, with the expectation of the model learning the pattern between instruction, input, and character-like output through fine-tuning.

**cl-tohoku/bert-base-japanese:**

For sentence embedding, I used cl-tohoku/bert-base-japanese, a pre-trained BERT model for Japanese text released by the Tohoku University NLP lab. As taught in class, BERT is a bidirectional encoder model that captures contextual meaning from both directions of a sentence. As a regular BERT model, this model has 12 layers, 768 hidden dimensions, and 12 attention heads. It was trained on the Japanese version of Wikipedia, which contains approximately 17 million sentences. The WordPiece tokenizer used in this model has a vocabulary size of 32,000. According to the documentation on Hugging Face, when tokenizing, texts will first tokenized by MeCab, a morphological analyzer, and then further split into sub words using WordPiece.

I used this model's tokenizer and embedding capabilities without fine-tuning. In the get\_bert\_embedding function, each sentence is first tokenized and then passed into the BERT model to obtain token-level embeddings. I then applied mean pooling over the token embeddings to obtain a fixed-size 768-dimensional vector for each sentence. These embeddings were later used for tone clustering (via K-means) and as inputs for the FFNN tone prediction model.

**FFNN:**

An FFNN is a simple neural network that processes inputs through fully connected layers and adjusts its weights using loss minimization to improve prediction accuracy. This model is used to predict how the character should respond to the user’s input. I used similar functions in homework 4 to create this model.

I used two datasets for training the FFNN: the J-CHAT dataset and the Chigiri-only dataset. I collected all unique tone labels from both datasets and assigned integer labels to them; these integers were used as the target outputs for training. Then, the sentences in each dataset would be embedded using BERT, and the result vectors would be fed into the FFNN model for training.

The model was trained with a batch size of 128 and a learning rate of 0.001. For the first stage of training with the J-CHAT dataset that provides a general emotional tone style across various speakers, the model was trained for six epochs. After finishing training the half model, I continued training the same model on the Chigiri dataset for an additional 20 epochs. I used more epochs for the second stage because the Chigiri dataset was way smaller, and capturing Chigiri's tone accurately was the main goal of this model.

**Rinna’s japanese-gpt-neox-3.6b-instruction-sft** and **japanese-gpt2-medium:**

GPT is a decoder-only transformer that can generate open-ended text based on the input. Originally, this project was intended to use rinna/japanese-gpt-neox-3.6b-instruction-sft for response generation, which accepts prompts separated by colons and the dialogue lines clearly indicate who is speaking with quotation marks. This model uses an input format with distinct speaker tags and newline symbols, making the format similar to natural conversation. However, with 36 layers and 2816 hidden units, the model was too large to be loaded on my device.

As a result, I switched to rinna/japanese-gpt2-medium released in 2021, which is a 24-layer, 1024-hidden-size transformer-based language model. Like the Tohoku BERT model, this GPT model was trained on Japanese CC-100 and Japanese Wikipedia. The model allows normal raw text as input and will generate text that continues the input. This behavior does not fully meet the needs of this project, as it simply continues the input text rather than generating a reply as a different speaker in a conversation. However, since my training data is structured as dialogues and language models can learn patterns from such data, it is reasonable to try to train a lightweight LoRA adapter and expect it would enable the model to behave like it is responding in a conversation.

I decided to use the LoRA technique to fine-tune the model on my datasets since the pre-trained model is too huge for me to train on all of the layers. Instead of updating all model parameters, LoRA works by adding a few small trainable matrices to certain layers, such as attention layers, so it can adapt to new data with much lower memory and compute costs. The two layers of the rinna/Japanese-gpt2-medium model that would be changed were the c\_proj (the concatenated attention projection layer) and the c\_attn (the concatenated attention output layer).

Similar to the FFNN model, I trained this model for two stages, first with the full *Blue Lock* dialogue and then with the Chigiri-specific dataset. For the first dataset, the two layers were trained for one epoch with a LoRA rank of 8, a scaling factor of 16, and a learning rate of 2e-4. I used these values, which would not make a huge impact on the model because there are too many characters with distinct personalities in the full data. The main task of this stage was to increase the chance of responses that are related to worldbuilding, such as soccer or strikers. For the second stage, the same model was trained with the Chigiri-specific dataset (including the description of tone) for eight epochs in total, with a LoRA rank of 8, a scaling factor of 32, and a learning rate of 5e-4. All the changes made here aimed to amplify the impact of this dataset on the model.

**What are the Results/Conclusions?**

Finally, with the FFNN model and the fine-tuned GPT model trained, I created a function that allows users to input text and print the response. The input would first be embedded by the Tohoku BERT model and then be sent to the FFNN to predict the Chigiri’s response tone. Together with the tone, the input would be used in Rinna’s GPT model to generate a response.

A screenshot of a computer program

AI-generated content may be incorrect.

After finishing creating the function, I compared the generations of my FFNN+fine-tuned GPT model, fine-tuned only the GPT model, and the original GPT model. From the result, I felt that after fine-tuning, the model's tone and phrasing became more aligned with the character’s world and personality, but the responses often felt off-topic.

**Original model vs. Fine-tuned model:**

The base model’s responses were generally coherent and well-structured; however, at the same time, it often focused on generic topics like games. The tone of the responses is also mostly positive and lacks characteristics. Although the formatting was clean and natural, the tone was too gentle and did not resemble the character’s speaking style in the anime.

On the other hand, the fine-tuned model produced responses that better reflected the world context (although the target character’s personality did not seem to be successfully simulated due to the lack of data). However, the response sometimes does not seem to be directly replying to the user’s input; for example, when the user asked, “What is your weapon?” the model would respond with, “I don’t think I can beat you.” One potential reason for this happening is that *Blue Lock*’s dialogue is often fragmented and delivered in short, back-and-forth exchanges; therefore, the pattern the model was going to learn would be similar, that the response doesn’t seem to be directly replying, but still somehow make sense. Additionally, I used custom tokens such as “##キャラ:” during training to indicate speaker roles, but this sometimes led to awkward formatting of repeated tags or misplaced tokens in the generated text.

**Generation with FFNN tone doesn’t meet the expectation:**

After comparing several generations with and without tone instructions, I noticed that the outputs without tone prompts were generally cleaner and less likely to contain unexplainable text. One possible reason is that the tone label “calm” (冷靜) appeared a lot in Chigiri’s training data, making the FFNN overly biased toward selecting it. As a result, the tone classifier consistently predicted the same label, leading to repetitive prompts like “冷靜に説明・論理的に助言・分析する口調で返事をください.”

Moreover, as the GPT2-medium model I used was not instruction-tuned, it did not treat special instructions (like tone descriptions) as separate inputs. To the model, the tone directive is just plain text. As most of Chigiri’s text has the tone of “calm,” there was not enough diversity for the model to actually realize the pattern of having “calm” in the beginning, resulting in a certain type of tone. Therefore, adding tone instructions only increased the prompt length, and this action likely introduced noise or distracted the attention from the actual message. This explains why the tone-prompted outputs often showed format errors or hallucinated tokens.

For future experimentation, I would definitely like to try on rinna/japanese-gpt-neox-3.6b-instruction-sft, which supports structured prompts and instruction-following, and to see if the result could be better than the one trained on this general model. Also, another thing that must be done to improve the performance is to find more well-preprocessed data for both dialogue and tone prediction. For the training data of the GPT model, the ideal future dataset should have well-separated sentences and complete sentences for each character. They could be both manually arranged or collected from more text-based works like novels. For the training data of FFNN, I believe it would work better if the base training data could be collected from Japanese instead of translated from English. I originally thought this difference would be acceptable if I gave the second dataset, the Chigiri-specified dataset, a larger learning rate and more epochs so it could learn about the Japanese style of reaction from it. However, from the result, I realized that this decision resulted in overfitting, especially when the Chigiri-specified dataset lacks diversity. Therefore, in the future, finding an applicable Japanese text with tone labeled would be important; I would be able to lower the influence of the Chigiri-specified dataset and only fine-tune it.

**Work Cited:**

Cl-Tohoku. (n.d.). CL-Tohoku/Bert-Japanese: Bert Models for Japanese text. GitHub. <https://github.com/cl-tohoku/bert-japanese>

Japanese-GPT2-medium: AI model details. japanese-gpt2-medium | AI Model Details. (n.d.). <https://www.aimodels.fyi/models/huggingFace/japanese-gpt2-medium-rinna>

Lora. LoRA. (n.d.). <https://huggingface.co/docs/peft/package_reference/lora>

Model temporarily unavailable. Your Roadmap to the AI Revolution - AIModels.fyi. (n.d.). <https://www.aimodels.fyi/models/huggingFace/japanese-gpt-neox-36b-instruction-sft-rinna>

Rinna/Japanese-GPT-neox-3.6b-instruction-sft · hugging face. rinna/japanese-gpt-neox-3.6b-instruction-sft · Hugging Face. (n.d.). <https://huggingface.co/rinna/japanese-gpt-neox-3.6b-instruction-sft>

Rinna/Japanese-GPT2-medium · hugging face. rinna/japanese-gpt2-medium · Hugging Face. (n.d.). <https://huggingface.co/rinna/japanese-gpt2-medium>

S. Poria, D. Hazarika, N. Majumder, G. Naik, E. Cambria, R. Mihalcea. MELD: A Multimodal Multi-Party Dataset for Emotion Recognition in Conversation. ACL 2019.

Tohoku-NLP/Bert-base-Japanese · hugging face. tohoku-nlp/bert-base-japanese · Hugging Face. (n.d.). <https://huggingface.co/tohoku-nlp/bert-base-japanese>