Model Training Method

GPT Fine-tuning

Despite the availability of later model releases such as ChatGPT-4 and the GPT-4 API, considering budget constraints, accessibility, and efficiency at the time of the initial model training, GPT-3.5 remains an ideal foundation for the Fine-tuning. This study employed a batch training method to feed GPT with multilingual corpora. The calibration of dialogue-specific fine-tuning of GPT facilitated nuanced understanding of linguistic cues within selected language conversations. Transfer learning techniques retain parameters that encode textual comprehension while adapting higher layers to the nuances of conversational context parsing. This study applied a transformer-based architecture to develop a contextual dialogue model, primarily by fine-tuning the GPT-3.5 model.

Once all data preprocessing was completed, the training data was segmented into batches and organized into a folder directory. These batches were subsequently divided into training, validation, and test portions, with a detailed explanation provided in the data training section. Both training and validation files were submitted to the fine-tuning job, with statistics provided throughout the training process, offering initial insights into the model's improvement. Additionally, a test set was utilized for evaluating the model output after training by generating samples on the test set. After completing these preparation steps, the data is ready to proceed through the GPT fine-tuning pipeline.

Regarding API specifications and restrictions at the time of model training, token[1] limits varied depending on the specific model. For this research prototype, the chosen model was gpt-

[1] A token is defined as a fundamental unit of text, typically a word or a sub word, used for various language processing tasks.

3.5-turbo-1106, which had a maximum context length of 16,385 tokens. This limit determined the size of each training batch. The newly released gpt-4 API was not open to the public and is limited for a small group of researchers to access.

To fine-tune the GPT model, it needs to learn from examples of successful chat conversations. These examples include user prompts and corresponding chat responses that are viewed as “correct answers” or golden sets. This process is equivalent to a teacher providing answers to exam questions for students to learn from. To perform one round of fine-tuning, at least 10 training examples were required. However, OpenAI suggests that using 50 to 100 training examples with gpt-3.5-turbo can lead to significant improvements in the model's performance.

The fine-tuning process for each batch involved several execution steps. The fine-tuning script, implemented in Python, utilizes the OpenAI API and comprises five fundamental steps:

1) It establishes an API connection and authenticates through the API.

2) It defines the conversation template, which encompasses user requests, the assistant's responses, and a system message specifying the AI's role in context parsing and interpretation.

3) It loads the training data into the prompt, converting it from Excel format to JSON format, rendering it ready for ingestion by the model.

4) It connects to the OpenAI API client and initiates the fine-tuning process, specifying parameters as instantiated (i.e., model=gpt-3.5-turbo-1106, n_epochs=3, batch_size=1, learning_rate_multiplier=2) or automatically chosen by the API.

5) It monitors the fine-tuning process by regularly checking the job status.

Once the fine-tuning job is completed, the custom GPT model is tested with a new prompt to verify the training results.

Semi-Supervised Training

For model training and development, this research fine-tuned a transformer-based architecture using both supervised and semi-supervised approaches. It's essential to note that GPT is already pretrained on a massive corpus of text data, providing a foundation for further customization. During training, previous model outputs were reviewed by human evaluators, with the corrected outputs being reintegrated into the dataset. This iterative process progressively refines the model's contextual understanding (see figure 9 below for the schematics of this process).

To maximize performance, especially in scenarios with limited labeled[2] data, the researcher explored semi-supervised learning, which combines elements of supervised and unsupervised techniques (Zhu, 2005). The process began with the supervised pre-training of CustomGPT on labeled data, where the model learns from explicitly labeled examples. Subsequently, I adopted sampling policies to alternate between supervised and unsupervised mini batches, drawing from both the labeled and pseudo-labeled[3] datasets.

Loss Functions and Training Details:

The supervised loss (Lsupervised) was calculated using cross-entropy:

$$L\text{supervised} = -\sum^{y} \log p\,(y|x)$$

In this equation, p(y|x) represents the predicted probability distribution over possible outputs (y) given the input (x) (Bishop, 2006). During each training iteration, the model's

[2] In this research, 'labeled' means data has been annotated and verified.
[3] In this research, 'pseudo-labeled' means that the labels are created by machine and have not undergone human review.

predicted probability distribution (p(y|x)) was compared to the ground truth (y) for each labeled instance (x). The supervised gradients were then computed based on the discrepancy between these distributions and the ground truth labels.

The unsupervised loss (Lunsupervised) incorporates a consistency regularization term:

$$L\text{unsupervised} = \left|\left|f(x) - f\left(T(x)\right)\right|\right|^2$$

In this equation, f(x) denotes the model representation of the input x, and T(x) applies a stochastic data augmentation function (Laine & Aila, 2016). The consistency regularization term measures the similarity between the model's representation of an input (f(x)) and that of its augmented version (f(T(x)). The unsupervised gradient, symbolized by the squared L2 norm (||...||^2), quantifies the dissimilarity between these representations and facilitates the alignment of the model's predictions with minor perturbations in the input (Ghosh & Thiery, 2021).

The total loss (Ltotal) was computed as a weighted sum of both supervised and unsupervised loss terms:

$$Ltotal = L\text{supervised} + \lambda L\text{unsupervised}$$

The parameter λ controls the balance between supervised and unsupervised learning components (Goodfellow et al., 2016), enabling experimental fine-tuning based on the chosen data split policy and the availability of labeled data.

The optimization of the model parameters involved stochastic gradient descent (SGD) techniques. Learning rates and other optimization hyperparameters played a significant role in the convergence and performance of the model. These values were adjusted based on empirical testing across different training phases.

Data Split Ratios

To optimize CustomGPT's performance, I conducted experiments with varying ratios of labeled to pseudo-labeled data.

1) 100% labeled data
2) 80% labeled data + 20% pseudo-labeled data

The semi-supervised learning process started with fully supervised pre-training, where the model relied on labeled transcripts. Then, predictions with ≥95% confidence were converted into pseudo-labels for semi-supervised learning, effectively augmenting the dataset. The labeled data was then redistributed following an 80:10:10 split for training, validation, and testing, respectively. Random sampling assigns labeled training data to these cohorts.

In summary, the two policies[4] were denoted as 100/0, 80/20. In the 80/20 data split policy, 48% of the data was designated to training, 16% to validation, 16% to testing, and 20% to unlabeled data (see table below).

Table 1. Data Split Policies

Training Mode	**Policy**	**Labeled (Train)**	**Labeled (Validation)**	**Labeled (Test)**	**Unlabeled**
Supervised	100/0	80%	10%	10%	0
Semi-Supervised	80/20	48%	16%	16%	20%

The learning workflow encompasses two modes: supervised for 5 epochs and semi-supervised for 3 epochs. Three data sources were involved: labeled transcripts, unlabeled

[4] A data split policy defines the strategy and proportions used to divide a dataset into subsets for training, validation, and testing.

transcripts, and pseudo-labeled transcripts, with pseudo-labeled transcripts starting at 0 and progressively increasing as the model became more adept at handling unlabeled data. A GPT custom model was created for each of the data split policies, with their performances evaluated using the test datasets.

The semi-supervised learning training plan is demonstrated in the diagram below:

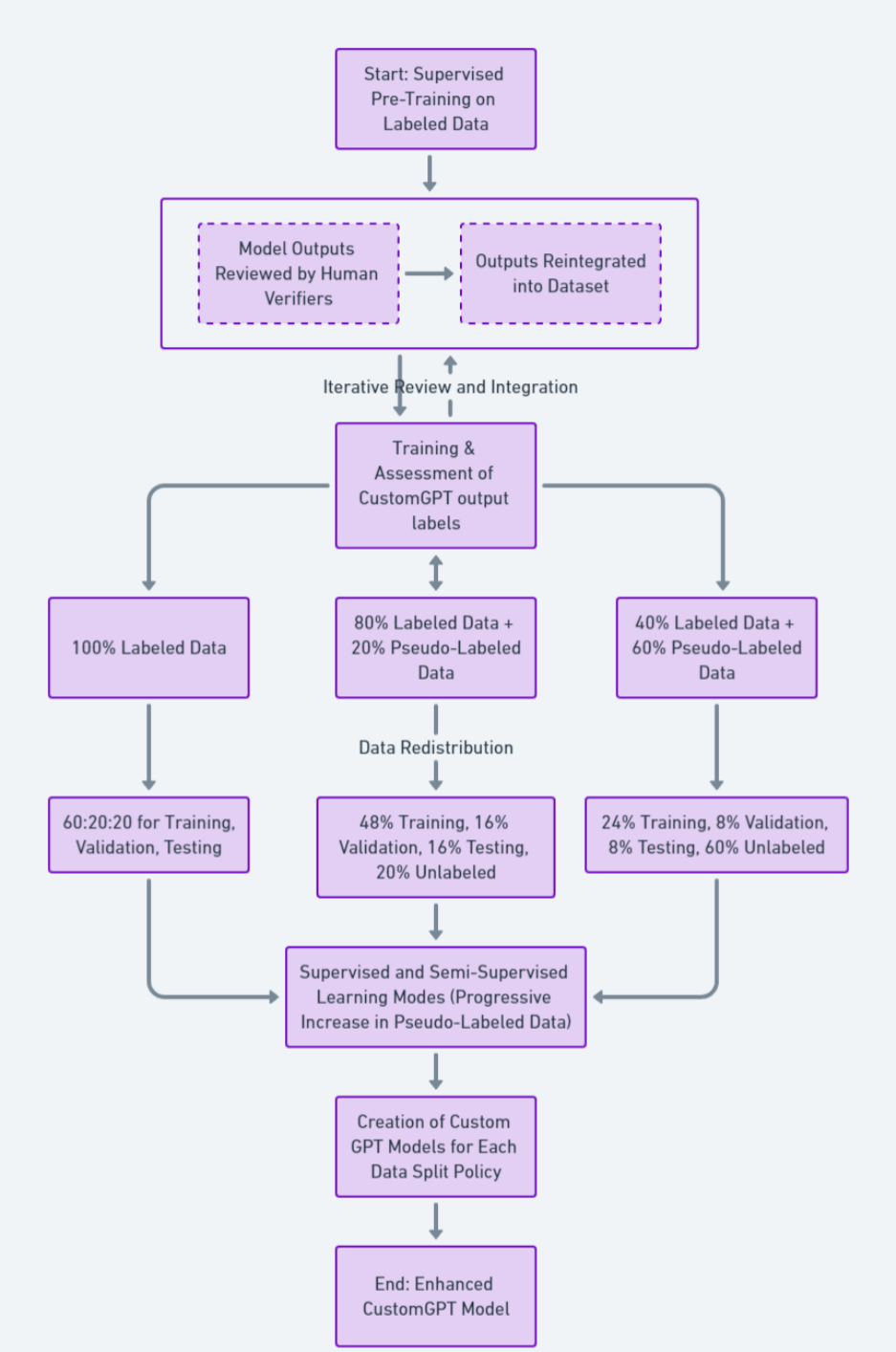

Figure 1. Semi-supervised Data Training Workflow

Model Fine-tuning Implementation

Base Model Selection

The proposed model heavily relies on the understanding and generation of human-like text across multiple contexts and languages. In this study, GPT-3.5, specifically the gpt-3.5-0125 model, was chosen as the base model due to its state-of-the-art performance in natural language understanding and generation tasks. GPT-3.5 encompasses models of various sizes, with the largest having 175 billion parameters. As a large-scale language model trained on a vast amount of diverse text data, GPT-3.5 can capture rich linguistic knowledge and generate coherent and contextually relevant responses.

This study primarily utilized GPT-3.5, updated as of January 25 of 2024, due to its accessibility and proven effectiveness. Despite the availability of GPT-4, it was not employed as a baseline because at the time of this writing, it was only accessible to a limited audience for experimental purposes. An application for access to GPT-4's fine-tuning capabilities was submitted; however, access had not been granted by the time of this study.

The discussion outlines the future plans for integrating GPT-4 and other advanced large language models (LLMs) as the foundation for future research. The transition to GPT-4 and similar advanced models is predicated on accessing these technologies once they become more widely available and receiving the necessary computational support to manage their increased complexity and size.

Learning Workflow

In this fine-tuning workflow, I created 9 different custom GPT Models by tweaking the numbers of epochs alongside with the learning rate multipliers. Two training modes are applied. Supervised models (100/0 policy split) using all transcription-annotation pairs with accuracy and sufficiency scores above averages and human feedback and corrections. These models are indicated by “supervise” or “rlhf” (i.e. Reinforcement Learning Human Feedback, indicating Supervised Mode) in the model's name.

This training mode mimic Reinforcement Learning Human Feedback but the human inputs are front loaded in the data selection process. The second training mode is semi-supervised learning with 80/20 policy split. These models are indicated by “full” in the model's name. The evaluated and approved labels are complemented by pseudo labels generated by the baseline models in the initial training.

Fine-tuning Setup

The fine-tuning process relies on several key parameters across training, validation, and testing phases. In addition to the preparation of training and validation files and the setup of GPT templates with user, system, and assistant role descriptions, several parameters define the outcome performance: temperature, top p, prompt, epoch, learning rate, and batch size.

Temperature, top p, and prompt influence the entire experiment, not just Fine-tuning, but also the baseline and test performance. To maintain the validity of the experiment, these parameters are kept consistent. The rationale for choosing specific values for temperature and top p is laid out in the discussion section, supported by side experiments and previous literature. Prompt engineering techniques are also discussed to explore future methods for enhancing the quality of the results. To streamline the experiment, a basic, instructional prompt is used without advanced techniques.

Epoch, learning rate, and batch size directly impact the Fine-tuning performance, in context with the number of samples/tokens. A detailed discussion on the rationale behind the chosen values for epoch, learning rate, and batch size, and supporting documentation, is also provided in the discussion section. Here, I provide a basic description of each component in the Fine-tuning configuration:

- ***Temperature: A*** hyperparameter that controls the randomness (creative vs. deterministic) of the model's output.

  Setup: In this experiment, a temperature of 0.3 is used

- ***Top p: A*** sampling technique that selects the next token from the top p most likely tokens based on the model's probability distribution.

  Setup: A top p value of 0.2 is employed in this experiment.

- ***Prompt:*** The prompt is the initial text input provided to the model, which sets the context and guides the generation process.

Setup: The prompts are designed to elicit relevant and coherent responses from the model in a required output format that facilitates further data post-processing.

Chinese version, user prompt: "{background} + **以上文作**为对话背景，帮助我注释以下的对白，分析一般背景，说话人的情绪基调，文化线索以及与世界构建的微妙联系，以表格格式将这4**个方面作**为附加的单元格将它们放在每**个**话语对应的单独列 + {batched transcripts}."

English version, user prompt: "{background} + using the above as a background for the dialogue, help me annotate the following dialogue, analyze the general context, emotional tone of the speaker, cultural clues and subtle connections to the world building, put these 4 aspects as additional cells in a table format and place them in separate columns for each utterance + {batched transcripts}. "

System prompt: "You are a helpful assistant. Your task is to analyze dialogue segments in its language for context, emotion and culture, based on the provided background.

An example in GPT playground format is included in the screenshot below (Figure 2). This is for illustration purposes only and the actual workflow is executed through Super GUI using Python 3.7 scripting language.

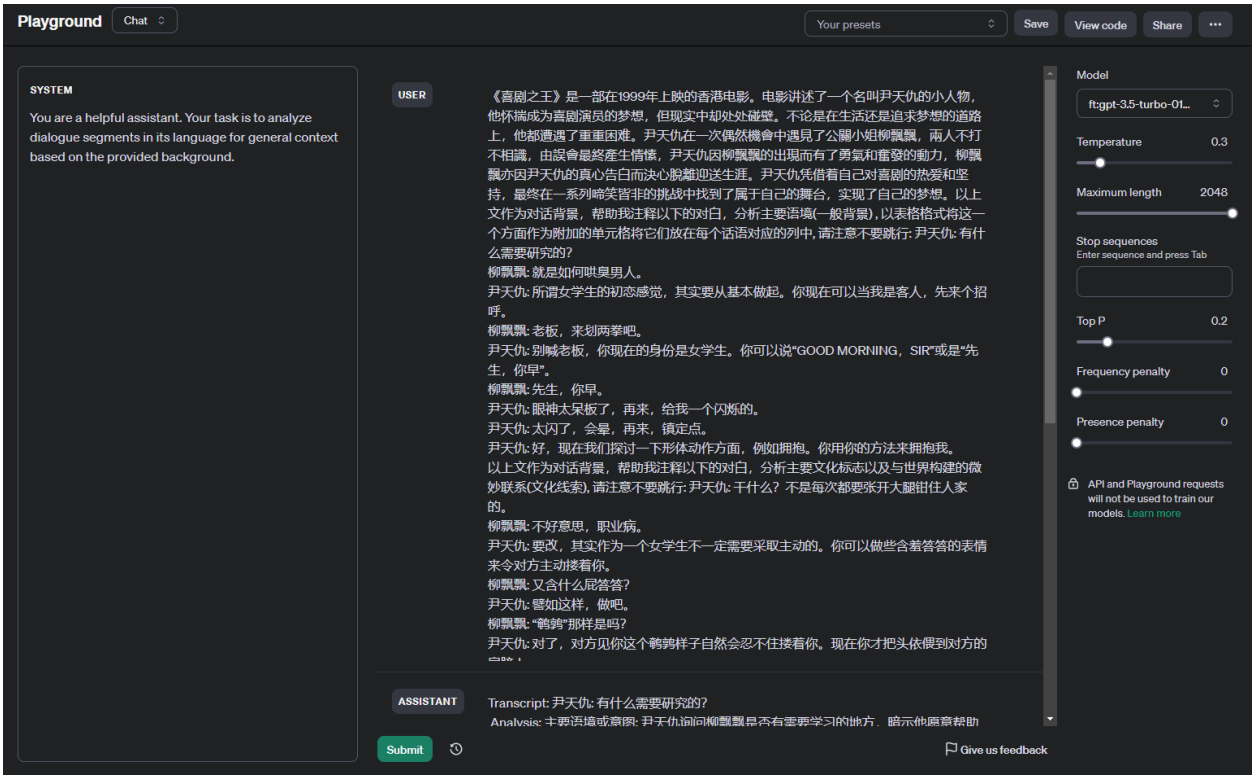

Figure 3. Prompt Template in GPT playground

- ***Epoch:*** An epoch refers to a single pass through the entire training dataset during the Fine-tuning process.

Setup: In this experiment, 3-8 epochs are used. The default ("Auto") Fine-tuning setting is 3 epochs.

- ***Learning rate:*** The learning rate is a hyperparameter that controls the step size at which the model's weights are updated during training.

Setup: Learning rate multipliers in the range of 0.5 - 2 are used in this experiment. The default ("Auto") Fine-tuning setting is 2.

- ***Batch size: R***efers to the number of training examples processed by the model in a single forward/backward pass.

Setup: A batch size of 1 is used because of the relatively small training size,

- ***Fine-tuning Template and Files:*** Defines the structure and format of the input data used during the Fine-tuning process. The template document is structured as conversations with a predefined system role, followed by a list of examples for the model to learn from. This structure is illustrated in the JSON format in figure 22.

Setup: JSON files are prepared with transcription-annotation sentence pairs for training and validation, with an approximate 8:1 split while maintaining dialogue integrity. For the supervised training mode, the examples are selected from annotations that have been deemed acceptable by human evaluators (scores higher than average). In the semi-supervised training mode, all examples are included, with unacceptable annotations corrected by evaluators.

```
{
  "messages": [
    {
      "role": "system",
      "content": "system instruction"
    },
    {
      "role": "user",
      "content": "user prompt"
    },
    {
      "role": "assistant",
      "content": "assistance response"
    }
  ]
}
```

Figure 4. JSON Object of "Conversation" for GPT Fine-tuning

Table 2. Fine-tuning Settings (Batch Size = 1)

Mode	Name	N train	N valid	Trained token	Runtime (h:mm)	Learning rate	Epoch
Supervised	supervise_ep4_rate1.2	68	11	203616	0:15	1.2	4
	supervise_ep4_rate1.5	68	11	203616	0:12	1.5	4
	supervise_ep5_rate2	68	11	254520	0:19	2	5
	supervise_auto_rate2	68	11	152712	0:11	2	3
	supervise_ep8_rate2	68	11	407232	0:28	2	8
Semi-Supervised	full_auto_rate2	144	18	357216	0:23	2	3
	full_ep8_rate0.1	144	18	952576	0:42	0.1	8
	full_ep4_rate0.5	144	18	476288	0:27	0.5	4
	full_ep5_rate0.5	144	18	595360	0:30	0.5	5