Exemplar Answer Generation with OpenAI API

Fine-Tuning LLM Evaluation Report

Content

Project Overview

This report focuses on the fine-tuning of a large language model(gpt-40-mini) to generate exemplar answers for educational purposes. The model has been customized using OpenAI's fine-tuning API and was trained with a dataset of student tasks, questions, rubrics, and reference answers. The goal of this project is to create a model capable of generating detailed and correct exemplar answers that align with assessment rubrics, aiding teachers in the evaluation of student responses.

The data used consists of question-answer pairs alongside the rubric criteria, structured into a JSON format, which was then transformed into JSONL to suit the OpenAI fine-tuning process. After splitting the dataset into training, validation, and testing dataset (70%, 15%, 15%), fine tunned the model using the training, validation dataset and subsequently evaluated its performance using multiple metrics.

Preparation and Processing

OpenAI Key Safety

The OpenAI key is saved into a .env file, and the .env file is added to .gitignore for safety reasons. This ensures that the key is not accidentally exposed in version control systems such as Git. Please place it in the root directory.

Installation Requirements

Environment: Python 3.9
To install the required packages, run this command in the terminal:

```
pip install -r requirements.txt
```

Since the project uses SpaCy for data preprocessing, you'll need to install the English model as well. Run this command in your terminal:

```
python -m spacy download en_core_web_sm
```

Main Files
There are two primary Jupyter notebooks to run:

1. **test.ipynb**: This notebook demonstrates the entire process of generating answers using the LLM (gpt-4o-mini and the fine-tuned model).
2. **evaluate.ipynb**: This notebook details the evaluation process, comparing the performance of the fine-tuned model with the baseline model (gpt-4o-mini).

Data Investigation

Attribute Name	Count
question_id	117
question	117
answer	117
task_id	59
task_title	59
task_content	59
rubric	115

After checking all the data, even though there is duplicate in rubric, there seems nothing wrong with the whole dataset.

JSON to JSONL Transformation

A Python script was created in the **test.ipynb** to transform the data into JSONL format. Each row included the prompt and completion structure expected by the fine-tuning API, ensuring compatibility with OpenAI's requirements.

Dataset Split

The data was divided into three parts: 70% for training, 15% for validation, and 15% for testing. The training set was used to fine-tune the model, the validation set monitored during training, and the test set for the final performance evaluation.

Token Usage Estimate Calculation

Using the tiktoken tokenizer to calculate the token used for the whole project.

Usage	Token
Fine Tunned Model Estimated	258,994
Fine Tunned Model Actual (by Open AI)	225,900
Train Dataset Token (18 questions)*2	46,662
Test Dataset Token (18 questions)*2	44,196
Validation Dataset Token (18 questions)*2	44,342
Total Estimate:	361,100

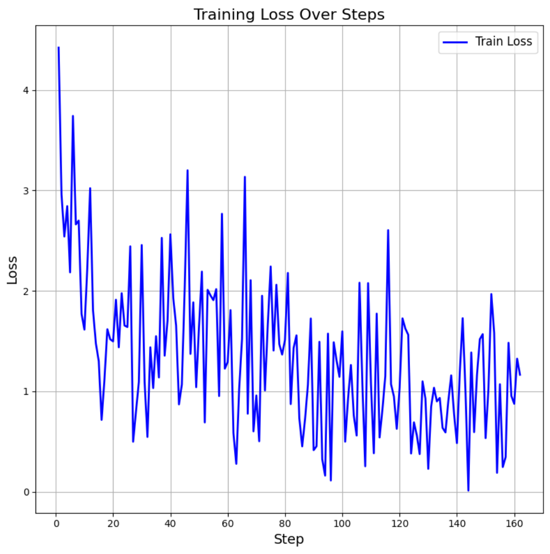
The model was fine-tuned over two epochs, which means that the total number of tokens used during the fine-tuning process should be multiplied by 2 to estimate the final count. The estimated number of tokens used in the training process was 225,900.

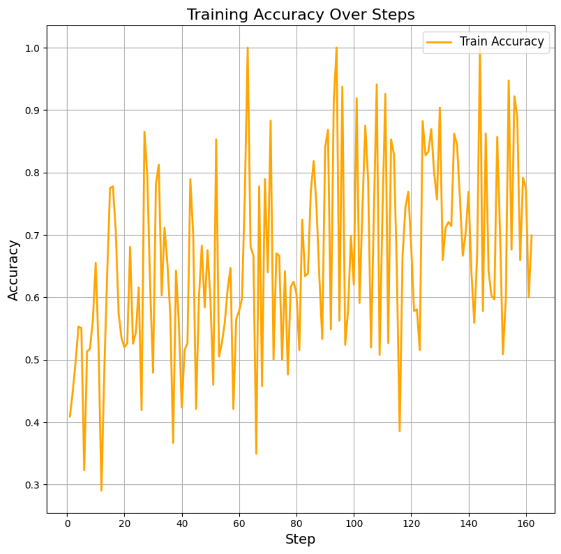
To ensure concise and informative responses, the length of the generated answers was limited to 260 tokens, based on empirical investigation and experimentation.

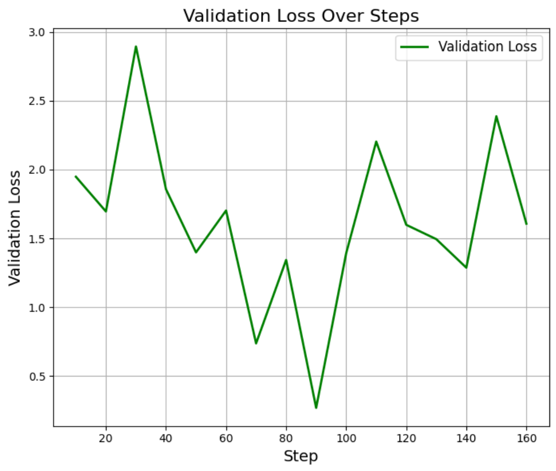
Model Training

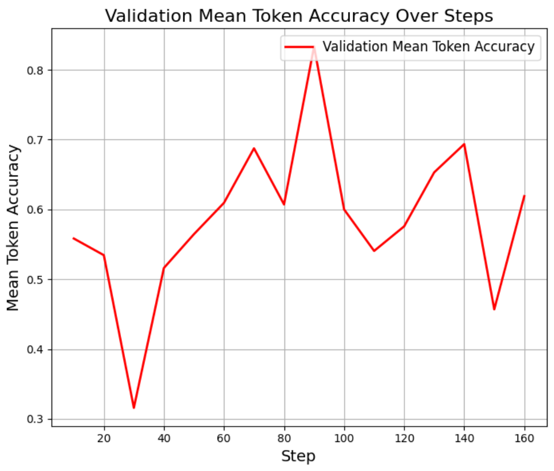
The fine-tuning process involved training the model with training data and evaluating its performance on the validation data. The OpenAI API was used to initiate the fine-tuning task. During training, validation loss was observed to determine if the model was learning generalizable patterns.

Model: \`gpt-4o-mini-2024-07-18\` was chosen. Hyperparameters were tuned, including \`n_epochs=2\`, to balance training duration and accuracy.









Training Graph Analysis

Training Loss: Shows an overall decreasing trend, indicating effective learning despite fluctuations.

Training Accuracy: Shows an increasing trend, suggesting improved understanding, with some variability due to training data.

Validation Loss: Fluctuates but remains stable, indicating balanced generalization without significant overfitting.

Validation Mean Token Accuracy: An upward trend suggests improving token-level predictions, though some fluctuations indicate varying data complexity.

Generate the Answer

Using the prompt to make sure all generated answer align with the requirements.
Generate exemplar answers for educational purposes based on a provided student task context, question, and rubric. The exemplar should:
Answer the question, using the information given in the context.
Meet the assessment rubric criteria for quality and correctness.
Be a reliable reference for teachers when evaluating students.
Be clear, concise, easy to follow, and short (max_completion_tokens = 260).

With settings， **max_completion_tokens** is 260 to make sure the answer should be precise.

The parameter **temperature=0.3** is used to control the level of randomness in the model's output during text generation.

A value like 0.3 is **Lower Temperature Values**, makes the output more deterministic, meaning the model is likely to choose the highest probability words, resulting in consistent and focused answers. This is particularly helpful in educational contexts where accuracy and reliability are crucial.

Evaluation

After training, the fine tunned model was compared against the original gpt-4o-mini and evaluated using key metrics to assess answer quality.

Data Cleaning and Preprocessing

Data Preprocessing Details, the following steps were taken:

Contraction Expansion: Common contractions (e.g., "can't" to "cannot") were expanded to ensure uniformity.

Tokenization and Lemmatization: Using SpaCy, each text was tokenized, lemmatized, and filtered to remove stopwords and punctuation, retaining only the meaningful content words. The cleaned text was then converted to lowercase for consistency.

Metrics Used

Three main evaluation metrics were applied to gauge the model's performance against the test set:

1. **BLEU Score**: Used to assess the degree of similarity between generated answers and reference answers based on n-gram overlap.
   This helped in understanding how accurately the model replicates specific phrasing from the reference texts.

2. **ROUGE Score**: Both ROUGE-1 and ROUGE-L metrics were calculated:

ROUGE-1 measures unigram overlap between the generated and reference texts, providing insight into the word-level content similarity.

ROUGE-L focuses on the Longest Common Subsequence (LCS), which evaluates how well the overall structure and logical flow match between generated and reference answers.

3. **BERTScore**: To evaluate the semantic similarity

Through leverages pre-trained BERT embeddings to determine how semantically aligned the generated answers are to the reference answers. This metric is particularly useful in ensuring the generated content captures not only the wording but also the intended meaning of the reference.

Evaluation Results

BLEU Score:

The model achieved an average BLEU Score of 0.2074 across the train and test sets in the Fine Tunned Model, indicating that the generated answers have limited phrasing overlap compared to reference answers. The lower mean BLEU score for the validation set suggests the potential for further improvements in generalization. But the fine-tuned model still has a better performance than the gpt-4o-mini, the base line model.

Fine Tunned	BLEU_train	BLEU_test	BLEU_validation
mean	0.207368	0.207368	0.073503
std	0.370137	0.370137	0.084942
min	0	0	0.000027

max	1.00	1.00	0.331808

Gpt-4o-mini	BLEU_train	BLEU_test	BLEU_validation
mean	0.022617	0.022617	0.022767
std	0.024128	0.024128	0.026662
min	0	0	0.002826
max	0.075166	0.075166	0.099555

ROUGE-1 and ROUGE-L:

The average ROUGE-1 Precision Score for Fine-Tunned Model was ` (0.566660 + 0.56666 + 0.532621)/3 = 0.555314`, and the average ROUGE-L Precision Score for Fine-Tunned Model was ` (0.505254 + 0.505254 + 0.486274)/3 = 0.4989273`.

BERTScore:
F1 Score:

Gpt-4o-mini	train	test	validation
mean	0.833929	0.833929	0.834965
std	0.035881	0.035881	0.019887
min	0.747745	0.747745	0.806073
max	0.885068	0.885068	0.865719

Fine Tunned	train	test	validation
mean	0.881423	0.881423	0.869037
std	0.064960	0.064960	0.027564
min	0.801404	0.801404	0.811929
max	1	1	0.929466

Mean F1 of Fine Tunned Model is 0.88, demonstrating high semantic similarity compared with the base model.

Analysis and Recommendations

The fine-tuned model outperforms the baseline (gpt-4o-mini) in BLEU, ROUGE, and BERTScore metrics, indicating better semantic and content understanding. However, lower scores on the validation or test set reveal that there is still room for improvement in model generalization.

Potential Improvements

- **Data Annotation**: Fine-tune the LLM using enhanced data annotation techniques. Like for each question, annotated all the important words or sentences for student to get the marks.
- **Human Review**: Ask Educators to review generated answers to ensure quality and relevance.

Conclusion

Fine-tuning improved BLEU, ROUGE, and BERTScore metrics, enhancing semantic alignment and adherence to rubric requirements. Further improvements in prompt engineering and data augmentation are recommended to boost generalization and model performance.