CMPSC 497 Final Project

Chih-Hsuan Sun

I. Introduction and Dataset Construction

i. Task Definition

The core task here is a type of **conditional text generation** with a large language model, where the model receives a concise input and generates a relevant answer. Specifically, the task designed for this project involves generating dedicated research approaches given a short problem description or topic (e.g., "intent classification with limited supervision").

ii. Data Collection and Curation

A custom dataset was constructed manually. Each sample includes:

- **Input:** A short text describing an NLP-related research problem (e.g., "entity linking in noisy texts")
- **Output:** A plausible and well-structured research approach to solve the given problem (generated with help of ChatGPT).

To ensure diversity and prevent overfitting, the dataset contains **100 samples** across various subfields of NLP including:

- Entity linking
- Summarization
- Dialogue systems
- Low-resource languages
- Bias detection
- Retrieval-augmented generation

An example of the sample would look like:

{"input": "entity linking in noisy texts",

"output": "Construct a joint model that aligns noisy surface forms with structured entities using contextual embeddings and edit-distance-aware alignment heads. Use weak supervision signals from Wikipedia anchors for initial training."}

iii. Formatting and Preprocessing

The following format was used to help the model understand the task structure. The inputs and outputs were then tokenized using AutoTokenizer with a maximum length of 512 tokens.

```
### Problem: <input>

### Approach: <output>
```

II. LLM Selection and Training Details

i. Model Selection

Two base models were explored in this project: **TinyMistral-248M-v3** and **Distilgpt2**. Both were selected based on their lightweight size and compatibility with limited hardware resources (single GPU with ~15GB VRAM on Colab).

Reasons for selecting these models:

- **TinyMistral-248M-v3** is a compact (~248M parameter) instruction-tuned model that allows for prompt-based finetuning and inference. Its small size made it suitable for full finetuning without exceeding VRAM limits.
- **DistilGPT2** is a distilled version of GPT2, widely used and well-supported. While not instruction-tuned by default, it offers stable performance and easy integration for causal language modeling tasks.

Side note: as far as I wanted to work with an actual LLM, it is too expensive and time consuming to finetune one. I simply do not have the resources to use 7B+ or even 2B models (I tried but was not successful).

ii. Training Setup

Fine-tuning was performed using Hugging Face's Trainer API with the following configuration:

- Epochs: 3
- Batch Size: 2
- Optimizer: AdamW (default in Trainer)
- Mixed Precision (fp16): Enabled
- Loss Function: Cross-entropy over next-token prediction
- Weight Decay: 0.01

III. Evaluation Metrics and Experiments

i. Evaluation Metrics

Two primary automatic evaluation metrics were used:

- **ROUGE-L**: Measures sequence overlap and structural similarity.
- **Cosine similarity score**: Uses contextual embeddings to assess semantic similarity between the generated answer and the reference ground truth.
- **Repetition Score**: A custom metric showing how often the model repeats itself (higher = more repetitive). This is included because the base model tends to have very high repetition in its response.

ii. Experiments

- **100-sample train set** and **10-sample test set** (constructed the same way as train set, with outputs as reference/ground truth) was used to prompt and assess performance.
- Inference was done in batch for efficiency.
- The **pretrained models** and the **fine-tuned models** for both **TinyMistral-248M** and **DistilGPT2** were evaluated under identical settings for direct comparison, with the former as baseline.
- Additionally, **LoRA fine-tuning** was attempted using the peft library. However, with large models the training failed on a 14.74 GB VRAM GPU due to CUDA out-of-memory errors, even with small batch sizes. As a result, only full fine-tuning was completed for this project.

IV. Experimental Results

i. Overall Performance

The comparison between the pretrained and fine-tuned TinyMistral-248M & DistilGPT2 models is shown in Table 1. This table clearly shows substantial improvements across both quantitative metrics and qualitative aspects of the generation after finetuning the model on my training data.

We see that:

- **ROUGE-L improved from 0.047 to 0.102** after fine-tuning DistilGPT2, and from **0.032 to 0.091** for TinyMistral-248M. This indicates that both models learned to better mirror the phrasing and structure of the reference answers.
- **Cosine Similarity increased from 0.288 to 0.525** for DistilGPT2 and from **0.157 to 0.420** for TinyMistral, suggesting a significant gain in semantic alignment between generated and reference outputs.

- **Repetition Score decreased from 1.5 to 1.0** for DistilGPT2 and from **7.62 to 1.0** for TinyMistral, showing that fine-tuned models produced more diverse, non-repetitive, and contextually grounded responses.

Metric	***Pretrained TinyMistral***	***Fine-Tuned TinyMistral***	***Pretrained DistilGPT2***	***Fine-Tuned DistilGPT2***
Average ROUGE-L	0.032	**0.091**	0.047	**0.102**
Average Cosine Similarity	0.157	**0.420**	0.288	**0.525**
Average Repetition Score	7.62 (looped)	**1.00** (clean)	1.50	**1.00** (clean)
Response Coherence	Off-topic, filler	On-topic, prompt-aligned	Generic, semi-coherent	Mostly task-aligned
Formatting & Structure	Broken or repetitive	Structured and fluent	Slightly generic	Reasonably fluent
Semantic Relevance	Poor	Conceptually consistent	Partial alignment	Closely aligned
Training Method	None	Finetuned on 100 samples	None	Finetuned on 100 samples

Table 1: Performance comparison post vs prior finetuning with TinyMistral-248M and DistilGPT2.

ii. Generation Examples

Sample generations along with performance from the test set are included in Table 2 and 3, with prior and post finetuning, respectively. From these results, we can conclude:

- **Pretrained Model** outputs were often unintelligible, off-topic, or entirely repetitive. For example, in the "Tokenization-free LM" case, the model looped variations of a generic phrase ("based on the idea of language learning") without contributing meaningful content.
- **Fine-Tuned Model** responses were concise, task-aligned, and structurally sound (at least to some extent). The model learned to address the prompt directly, producing structured research approaches even within the same token budget.

Prompt	Model	Generation (Truncated)	ROUGE-L	Cosine Sim	Repetition Score
Topic drift in conversations	TinyMistral (Pre)	*## Introduction...* (repeated filler)	0.019	0.149	4.67
Answer reranking in hybrid search	TinyMistral (Pre)	*I'm trying to solve a problem in R...* (repeated 8x)	0.033	-0.024	5.50
Tokenization-free LM	TinyMistral (Pre)	*...based on the idea of language learning...* (looped)	0.000	0.068	14.33
Topic drift in conversations	DistilGPT2 (Pre)	*To find a solution for these issues...*	0.050	0.232	1.83
Answer reranking in hybrid search	DistilGPT2 (Pre)	*Reranking is not so easy for most search engines...*	0.062	0.480	1.00
Tokenization-free LM	DistilGPT2 (Pre)	*The problem is that the tokenization solution...*	0.015	0.198	1.00

Table 2: Generation examples and performance with pretrained TinyMistral-248M and DistilGPT2, prior to finetuning.

Prompt	Model	Generation (Truncated)	ROUGE-L	Cosine Sim	Repetition Score
Topic drift in conversations	TinyMistral (Fine-Tuned)	*Develop a pipeline with shared topic drift modules...*	0.075	0.440	1.00
Answer reranking in hybrid search	TinyMistral (Fine-Tuned)	*Train a hybrid reranker using reranker queries...*	0.100	0.519	1.00
Tokenization-free LM	TinyMistral (Fine-Tuned)	*Develop a tokenization-free model using...*	0.056	0.274	1.00

Topic drift in conversations	DistilGPT2 (Fine-Tuned)	*Introduce a framework for summarizing topic drift...*	0.078	0.506	1.00
Answer reranking in hybrid search	DistilGPT2 (Fine-Tuned)	*Introduce a hybrid filter function that pairs queries...*	0.138	0.501	1.00
Tokenization-free LM	DistilGPT2 (Fine-Tuned)	*Use tokenization-aware tokenization using multilingual...*	0.114	0.377	1.00

Table 3: Generation examples and performance with finetuned TinyMistral-248M and DistilGPT2.

iii. Notes on the Results

On top of the shown results, I have also attempted training LoRA with TinyMistral-248M, which however produces performance similar to the original pretrained model, so I did not include the results with LoRA here.

While I managed to improve model performance on our test set via fine-tuning, note that as we have very limited sample size, the improvement does not mean the answers are correct, just comparatively much better aligned with our reference outputs. The models (inevitably) remain heavily reliant on fixed prompt formats, and perform relatively well only when inputs closely resemble the training data, but will struggle with unseen or loosely structured prompts (limited generalization due to model size and data size), leading to potential hallucinations.

Also, some fine-tuned outputs, while more on-topic, still lacked depth or prolonged reasoning, which is a common challenge of small-scale language models in open-ended tasks.

V. Challenges and Take Aways

- **Overfitting & Output Repetition**: With only 100 training samples, the model would often repeat sentence structures or phrases, caused by overfitting and can sometimes mislead evaluation. I had to carefully ensure the diversity and quality of the inputs and especially outputs in both train set and test set to reduce this effect. Out of everything, creating data was in fact the most time consuming for me on the task.
- **Prompt Sensitivity**: Minor changes in prompt phrasing (e.g., adding punctuation) led to large variations in generation, revealing the model's high sensitivity to input formatting.
- **Prompt Engineering Matters**: Adopting a structured format (### Problem: → ### Approach:) helped guide the model during fine-tuning. The prompt design played a significant role in maintaining alignment between training and evaluation.

- **LoRA Setup & Limitations:** A LoRA configuration was set up using peft. However, due to limited VRAM (15 GB), even lightweight models faced CUDA OOM errors during training. While LoRA reduced trainable parameters to under 0.04%, performance did not improve significantly, possibly due to undertraining or batch size constraints.
- **Training Costs Are High**: I was not able to fine-tune an actual "large" language model since it simply takes too much VRAM. Even for lightweight models like TinyMistral, prolonged training consumed significant GPU memory and compute units.
- **Training Data is the Bottleneck**: Due to time constraint (explained below), I did not have time to generate enough data (only 100 for train and 10 for test). For this task I believe to have a model that performs reasonably well, I believe I will need at least a few thousands of good quality samples.

Additional Comments

I would like to note that I was only able to spend very little time working on this specific project topic, because for the longest time **I thought our final project topic would be the one that was assigned to us in-class, during the final project experiment lectures, which I wrote a proposal for, and actually spent time researching into and working on for weeks.** I did not realize that the actual topic released later on is a completely unrelated one, until I found the project document right before the deadline, a mistake of mine.

I will still share my Colab work for the topic **"POS tagging for endangered language"** here, since I do not want my time spent on it wasted.

https://colab.research.google.com/drive/1k5_mu2MM3iYJKRZ9IQdIZAP9kct2wzYn?usp=sharing

Conclusion

With this project, I aim to demonstrate the viability of domain-specific fine-tuning for structured generation tasks, with relatively small models and datasets. Standard fine-tuning on smaller LLMs like TinyMistral or DistilGPT2 can be feasible and effective for focused tasks. Unfortunately, I did not have enough time or resources to spend on the project as I wish to, specifically on generalization.

For future direction, I would consider scaling the dataset to at least 1,000 samples (or even extracting data from available APIs), incorporating contrastive loss, and refining decoding strategies (e.g., top-k sampling + penalty tuning) to achieve a better generalizable model.