CS 613: NLP

Assignment 3: Fine Tuning & Evaluation

Heer Kubadia (22110096) | Jiya Desai (22110107) | Lavanya Lavanya (22110130) |
Shrishti Mishra (22110246) | Utkarsh Srivastava (22110278)

23 November, 2024

Introduction

The objective of this assignment is to fine-tune a pre-trained model (Llama 3.2-1B) for two specific tasks: **Text Classification (SST-2)** and **Question-Answering (SQuAD)**. Performance metrics were evaluated on the test set before (zero-shot) and after fine-tuning the model.

Model Selection

Model Selected: Llama 3.2-1B

- **Number of Parameters:**
  The model was calculated to have approximately **1.23 billion `(1,235,814,400)` parameters** , as confirmed by counting the parameters programmatically. This matches the number reported in the official Llama 3.2-1B documentation.

Fine-Tuning Details

- **Dataset Splitting:**
  - Train-Test Split: 80:20
  - Sampling: Stratified random sampling
  - Seed: 1
- **Fine-Tuning Tasks:**
  - **Task 1: Classification (SST-2 dataset)**

*The **LLaMA model**, as a language model, is primarily designed to predict the next token in a sequence and does not inherently include mechanisms for classification tasks such as SST-2 sentiment analysis.*

*To enable the model to perform classification tasks, we must add a **classification head** (a simple linear layer) on top of the contextual embeddings generated by the model. This classification head maps embeddings (usually derived from the `[CLS]` token or an equivalent) to logits corresponding to the target classes. The logits can then be used to compute the classification loss during fine-tuning.*

Also during fine-tuning, we froze all the layers of the base model and only fine-tuned the classification layer. This approach was chosen to optimize memory usage and computation resources, ensuring efficient fine-tuning without modifying the base model's parameters.

- **Task 2: Question-Answering (SQuAD dataset)**

*The **LLaMA model**, as a language model, is primarily designed to predict the next token in a sequence and does not inherently include mechanisms for tasks like question answering (e.g., SQuAD) either.*

To enable the model to perform question-answering tasks, we add a task-specific head on top of the model's contextual embeddings. For SQuAD-style tasks, this head typically includes two linear layers: one to predict the start position and another for the end position of the answer span within the context. These layers map the contextual embeddings of the input sequence to logits for the start and end indices, which are then used to compute the span extraction loss during fine-tuning.

During fine-tuning, we froze all the layers of the base model and only fine-tuned the task-specific head. This approach was chosen to optimize memory usage and computation resources, ensuring efficient fine-tuning without modifying the base model's parameters.

Metrics Evaluated:

- **Classification**: Accuracy, Precision, Recall, F1 Score
- **Question-Answering**: Squad_v2, F1, METEOR, BLEU, ROUGE, Exact Match

---

Results and Analysis

- **Task 1: Text Classification for Sentiment Analysis (SST-2)**

After fine-tuning the model for three epochs:

Epoch	Training Loss	Validation Loss	Accuracy	Precision	Recall	F1
1	0.555700	0.604033	0.819954	0.815385	0.835586	0.825362
2	0.606700	0.543783	0.848624	0.873206	0.822072	0.846868
3	0.505800	0.527815	0.857798	0.873832	0.842342	0.857798

Comparison before and after fine-tuning done.

Metric	Before Fine-Tuning	After Fine-Tuning
eval_accuracy	0.4415	0.857798
eval_precision	0.553191	0.873832
eval_recall	0.00690296	0.842342
eval_f1	0.0136358	0.857798

Analysis:

- Fine-tuning significantly improved all metrics.
- Zero-shot learning achieved decent scores, as Llama 3.2-1B is pre-trained on diverse datasets, but lacked task-specific contextual understanding.
- Fine-tuning enabled the model to capture sentiment-specific patterns, resulting in high accuracy and balanced precision/recall.

- **Task 2: Question-Answering (SQuAD)**

Metric	Sub-Metric	Before Fine-tuning	After Fine-tuning
squad_v2	HasAns_exact	1.45	27
	HasAns_f1	3.98	30.91
	HasAns_total	2000	500
	best_exact	1.54	31.2
	best_exact_thresh	0	0
	best_f1	4.01	32.25
	best_f1_thresh	0	0
	exact	1.55	29.67
	f1	4.19	31.39
	total	2000	500
bleu	-	0.0036	0.0039
rouge	rouge1	0.0271	0.011
	rougeL	0.0284	0.0156
	rouge2	0.0132	0.0063
meteor	-	0.067	0.0345
f1	-	0.032	0.3254
exact_match	-	0.0155	0.2967

The suboptimal results, even after fine-tuning, are primarily due to training only the top layer of the model's parameters. This approach was necessitated by memory constraints; attempting to train on all parameters resulted in system crashes due to insufficient memory. As a result, we had to limit the training to a smaller set of parameters to accommodate these hardware limitations.

Analysis:

- Fine-tuning led to slight improvements in all metrics.
- The zero-shot performance indicates the pre-trained model's general ability to generate answers, albeit with lower accuracy and relevance.
- Fine-tuning helped refine the model's ability to understand and generate contextually accurate answers, improving the evaluation scores.

Parameter Analysis

- Pre-trained Model Parameters: **1235814400**

- After adding a classification head and fine tuning for classification task,

  Fine-tuned Model Parameters: **1235818496**

The base model has a fixed number of parameters designed for language modeling tasks. Fine-tuning adjusts the values of these parameters for task-specific adaptation but does not alter the architecture or parameter count. The increase in the total parameter count occurs due to the addition of the classification layer, which introduces new weights in the form of a linear layer. Specifically, the classification head adds 4,096 parameters to the base model's parameter count. After adding this layer, the total number of parameters remains constant during fine-tuning, as fine-tuning only updates the existing parameters (including those in the classification layer) to adapt the model to the specific task.

- After adding a question-answering head and fine tuning for question answering task,

  Fine-tuned Model Parameters: **1235818498**

The base model has a fixed number of parameters designed for language modeling tasks. Fine-tuning adjusts the values of these parameters for task-specific adaptation but does not alter the architecture or parameter count. The increase in the total parameter count occurs due to the addition of the question-answering head, which introduces new weights in the form of two linear layers: one for predicting the start position and another for the end position of the answer span.

Specifically, the question-answering head adds 4,098 parameters to the model. This accounts for 2×(hidden size+1) , where the extra 2 parameters come from the bias terms in each of the two linear layers.

After adding this head, the total number of parameters remains constant during fine-tuning, as fine-tuning only updates the existing parameters (including those in the QA head) to adapt the model to the specific question-answering task.

Observations and Rationale

- **Lower/Higher Scores:**
  - Zero-shot scores were lower due to the lack of task-specific tuning.
  - Fine-tuning optimized the model's understanding of task-specific datasets, resulting in significant improvements.
- **Parameter Understanding:**
  - Fine-tuning adjusts pre-trained weights rather than increasing model complexity, which is why parameter counts remain consistent.
- **Performance Differences:**
  - Zero-shot: General knowledge from pre-training provides a baseline but lacks specific task adaptation.
  - Fine-tuned: Customization for SST-2 and SQuAD tasks enables better feature extraction and decision-making for respective tasks.

Model Upload

The fine-tuned model has been uploaded to 🤗 Hugging Face, ensuring reproducibility and accessibility for further evaluation. You can access the model with the attached links -

- fine_tuned_llama_SST-2
- fine_tuned_llama_SQuAD

Contributions

1. Heer: Fine-tuned the model on the SST-2 dataset.
2. Jiya: Fine-tuned the model on the SQuAD dataset.
3. Lavanya: Analyzed performance metrics and documented the results.
4. Shrishti: Implemented the code to calculate the number of parameters , verified consistency with the model's documentation and drafted the report.
5. Utkarsh: Conducted train-test split and ensured stratified sampling with the correct seed and drafted the report

References

- Llama 3.2-1B official documentation
- SST-2 Dataset Description
- SQuAD Dataset Description