CS7.501: Advanced NLP — Assignment 4

Instructor: Manish Shrivastava

Deadline: Nov 15, 2023 — 23:59

1 General Instructions

1. You should implement the assignment in Python.
2. Ensure that the submitted assignment is your original work. Please do not copy any part from any source, including your friends, seniors, and/or the internet. If any such attempt is caught, serious actions, including an F grade in the course, are possible.
3. A single .zip file needs to be uploaded to the Courses Portal.
4. Your grade will depend on the correctness of answers and output. Due consideration will also be given to the clarity and details of your answers and the legibility and structure of your code.
5. Please start early to meet the deadline. Late submissions won't be evaluated.

2 Objective

Implement prompt tuning on a GPT-2 small model using PyTorch and fine-tune it on three tasks: summarization, question answering, and machine translation. You can read the original paper to get a better understanding:

2.1 Setup

- Use the `transformers` library from Hugging Face to load the GPT-2 small model and tokenizer.
- Ensure you have the necessary datasets for each task.

2.2 Architecture

2.2.1 Soft Prompt Embedding Layer

Create an embedding layer for the soft prompts. This layer will have its own parameters separate from the GPT-2 model. The size of this embedding layer should be `[num_prompts, embedding_size]`, where `num_prompts` is the number of tokens in your soft prompt and `embedding_size` is 768 for the GPT-2 small model.

2.2.2 Model with Soft Prompt

Modify the GPT-2 model to accept the soft prompt embeddings and concatenate them at the beginning of the input sequence.

2.2.3 Fine-tuning Architecture and Hyperparameters

For the fine-tuning process, consider the following architecture and hyperparameters:

- **Batch Size:** Can be anything. If facing memory constraints, consider using gradient accumulation as a strategy to effectively increase the batch size without requiring additional memory.
- **Epochs:** Train for up to 10 epochs for each task, but consider early stopping based on validation loss to prevent overfitting.
- **Gradient Clipping:** Clip gradients with a norm of 1.0 to prevent exploding gradients.

2.3 Fine-tuning on Tasks

2.3.1 Summarization

- Dataset: Use the CNN/Daily Mail dataset. Link
- Soft Prompt: Initialize with tokens like `[SUMMARIZE]`.

2.3.2 Question Answering

- Dataset: Use the SQuAD dataset. Link
- Soft Prompt: Initialize with tokens like `[QUESTION]`, `[ANSWER]`.

2.3.3 Machine Translation (e.g., English to German)

- Dataset: Use the Europarl dataset. Link
- Soft Prompt: Initialize with tokens like `[TRANSLATE_EN_DE]`.

2.4 Training

- During training, prepend the soft prompt tokens to your input sequences and backpropagate only through the soft prompt embeddings, keeping the GPT-2 parameters frozen.

2.5 Evaluation

Evaluate the performance of your fine-tuned model on each task using appropriate metrics.

3 Questions & Grading

[Total marks: 100, Bonus marks: 25]

3.1 Theory [20]

1. **Concept of Soft Prompts:**
   How does the introduction of "soft prompts" address the limitations of discrete text prompts in large language models? Why might soft prompts be considered a more flexible and efficient approach for task-specific conditioning?

2. **Scaling and Efficiency in Prompt Tuning:**
   How does the efficiency of prompt tuning relate to the scale of the language model? Discuss the implications of this relationship for future developments in large-scale language models and their adaptability to specific tasks.

3.2 Implementation & Training [80]

1. **Architecture [30 marks]** Follow the architecture as described in the section above. Make sure to implement the Soft Embedding layer correctly, and then accordingly modify the GPT-2 model to incorporate soft prompts.

2. **Finetuning [35 marks]** Finetune on the three tasks as described above, with the appropriate datasets and prompt initialization.

3. **Evaluation and Results [15 marks]** Report relevant metrics attained on the tasks as mentioned above. Store the results in terms of output and put them in your report, which will be taken into consideration if your model does not perform well on the metrics.

3.3 Analysis [10]

Write up sufficient analysis on the performance of your model in a properly compiled report. This includes the hyperparameters you selected in training (like loss functions, # epochs, learning rate, optimizer, etc.) and metrics (for the finetuning). Report all the metrics, and store the results of the model seperately.

3.4 Presentation [10]

We would check these points at the time of individual evaluations that would involve code walk-through, explanation of the report analysis, and questions based on the implementation.

- Implementation efficiency & quality
- Report quality
- Inclusion of a readme along with the code
- Crisp & clear explanation of the code during evals

3.5 Bonus: Soft Prompt vs. Hard Prompt [25]

Investigate the effectiveness of learned soft prompts in comparison to manually designed hard (discrete) prompts. Begin by designing discrete text prompts for each task and use them to condition the model. Evaluate the model's performance when using these hard prompts and contrast it with the results achieved using soft prompts. Report the results and your analysis on this task.

4 Submission format

Zip the following into one file and submit it on the Moodle course portal:

1. Source code (Should include .py files only.)
2. Finetuned model
3. Report answering all the questions in PDF format
4. Readme file (should contain instructions on how to execute the code, restore the pre-trained model, and any other information necessary for clarity)

   Note: If the model size crosses the file size limits, upload them to your OneDrive folder and share the read-only accessible links in the readme file.

5 FAQs

1. Can I reduce size of dataset or number of epochs due to computational constraints?
   Yes, you can. However, make sure to mention this in your report, with appropriate justification for your choices.
2. Do I need to use lemmatization as a part of my data preprocessing?
   No, lemmatization is not necessary for this assignment.

3. Can I use tools like spaCy, torchtext, etc. for data cleaning and preparation?
   Yes, you are encouraged to use such tools for ease in the data preparation process. Focus of the assignment is on the core implementation and training using the implemented architecture.

4. Is it okay to include the test data in the pretraining process?
   No, going by the general ethics in machine learning, data used for evaluation should be something completely unknown to the model. You may make use of the train data or any other data not used in evaluation for the pretraining process.

5. How should the analysis be like?
   We have added the primary points you need to add in Section 3.3. You are free to develop the analysis section based on these pointers. Try to make it as descriptive and well documented as possible, giving the reader a clear idea on the experiment(s) you've tried and the results obtained. Kindly note that presentation holds its weight in the grading as well.

6. Can I submit my code in Jupyter Notebooks?
   No, the final submission should be a Python script. You may work using Jupyter Notebooks, but make sure to convert them to .py files before submitting.

6 Resources

6.1 Papers

1. Language Models are Unsupervised Multitask Learners

2. The Power of Scale for Parameter-Efficient Prompt Tuning

6.2 Explanatory Supplements

1. Brief Review — The Power of Scale for Parameter-Efficient Prompt Tuning

2. Soft Prompts

3. Prompting

4. Prompt Tuning, Hard Prompts & Soft Prompts