Default Final Project: GPT model and Downstream Tasks

April 2025

1 Overview

In this project, you will build the GPT model and use it to complete several downstream tasks. The project is divided into three parts:

- Implementation of GPT algorithms;
- Training and evaluation of a mini-GPT model;
- Execution of downstream tasks.

2 Implementation of the GPT algorithms

In this section, you are required to implement the core algorithms to train a GPT model, including the attention strategies, the GPT layer, the optimizer, and the fine-tuning algorithm.

For this project, you will need to load the pre-trained weights of the GPT-2 model for downstream tasks. Therefore, your implementation should be compatible with the official implementation. A good starting point is the codebase available at this link, which sets up the workflow and specifies where you should implement the algorithm. This codebase is adapted from the final project of Stanford CS224n (historically adapted from the “minbert” assignment developed for Carnegie Mellon University’s CS11-711). For more details about the codebase, refer to Part 4 of this document.

Alternatively, you can implement your own version of the GPT model without using the framework mentioned above. However, ensure that your implementation correctly loads the weights of the pre-trained GPT-2 model. You can review the code for loading GPT-2 model weights in the framework at this link to better understand the process. Please note that your implementation should involve at least as much algorithmic detail as required by the above framework.

If you use the above framework, you can run the testing scripts provided in the codebase to verify the correctness of your implementation after implementing the algorithm. For more details, see the homepage of the codebase.

3 Training and evaluating a mini-GPT model

In this task, you will train a mini-GPT model to generate text in the style of William Shakespeare.

- **Dataset Preparation:** Use the dataset ”downstream-tasks/data/tinyshakespeare.txt” from the codebase as the source dataset. Split the dataset into training, validation, and testing sets with a ratio of 80%, 10%, and 10%, respectively.
- **Training:** Train the GPT model from scratch without loading any pre-trained transformer weights (word embeddings are allowed). Experiment with at least two different network architectures and hyperparameter settings to observe the impact of each on the training procedure.
- **Model Selection:** Monitor the performance (loss or perplexity) on the validation set to select the best model for each network architecture and hyperparameter configuration.
- **Performance Evaluation:** Evaluate the performance of the selected model on the test set. Report the evaluation metrics and provide several generated samples for each model.

4 Execution of downstream tasks

In this section, you are required to accomplish three downstream tasks: 1) Sentiment Analysis, 2) Paraphrase Identification, and 3) Sonnet Generation.

For each task, complete the following steps:

- **Resource Identification:** Locate the dataset and example code in the ”downstream-tasks” folder of the codebase; the details about the datasets and tasks are also described in Part 6 and Part 7 of this document.
- **Algorithm Implementation:** Implement the fine-tuning algorithm by either completing the code in the example or writing your own implementation;
- **Model Training:** Load the pre-trained GPT-2 model weights into your model, and apply your fine-tuning algorithm to the given dataset to train a task-specific model for each task. Use the validation/development set to select the best model;
- **Performance Report:** Provide a performance report on the development set, analyzing any poor-performing cases. Generate prediction results for both the development and test sets, and submit them to the system. The format of the prediction results should match that of the development dataset file (with the ground-truth labels replaced by the predicted labels) and the test dataset file (with the predicted labels added).

Strive to achieve the best possible performance on these tasks.