Homework 5: Build Your Own LLM

Eman Ansar
eansar@andrew.cmu.edu
Carnegie Mellon University

Olabode Ajenifujah
oajenifu@andrew.cmu.edu
Carnegie Mellon University

Siddharth Saha
ssaha3@andrew.cmu.edu
Carnegie Mellon University

Yujie Li
yujiel2@andrew.cmu.edu
Carnegie Mellon University

***Abstract*—This report provides performance of our Large Language Model which was pretrained on a subset of OpenWebText data, and fineturned on three tasks which includes summarization, question answering and sentiment analysis. We adopted a version of GPT-2 small, a decoder-only variant of the Transformer architecture. This selection rationale for a decoder-only approach is elucidated, emphasizing its suitability for few-shot learning scenarios and robustness in generating meaningful outputs. Furthermore, the report outlines the fine-tuning possible process for task-specific datasets, by introducing a special token to delineate context and result for tasks such as summarization, question answering, sentiment analysis, and named entity recognition. The code and weight files are also made available.** [1]

I. Introduction

Language models have emerged as foundational tools in natural language processing, enabling machines to understand and generate human-like text. Transformer-based models, such as BERT [1] and GPT [2], have particularly revolutionized the field, demonstrating unparalleled performance across various tasks. However, their widespread adoption is often hindered by their computational demands, limiting their deployment in resource-constrained environments.

To address this challenge, this study introduces a novel approach to building a Lightweight Language Model (LLM) by leveraging a decoder-only architecture and task-specific fine-tuning. Unlike traditional transformer models that consist of both encoder and decoder components, the LLM architecture employs only the decoder, significantly reducing computational complexity while preserving performance. The pretraining corpus for the LLM is constructed using a carefully curated subset of the OpenWebText dataset ensuring a diverse and representative collection of text sources. By focusing on the decoder component, our approach capitalizes on the model's generation capabilities while minimizing computational overhead.

Furthermore, we investigate the efficacy of task-specific fine-tuning on three key downstream tasks: summarization, question answering, and sentiment analysis. Through this fine-tuning process, the LLM learns task-specific representations, thereby enhancing its adaptability and performance across diverse applications.

In this paper, we present the methodology for constructing the LLM, detail the fine-tuning process for each task, and evaluate its performance against established benchmarks. Our experimental results demonstrate the effectiveness of the proposed approach in achieving competitive performance on various tasks while requiring substantially fewer computational resources compared to traditional transformer models.

[1]Drive link: https://shorturl.at/hsLMP

II. Literature Review

The use of LLMs in natural language processing (NLP) has revolutionized how machines understand and generate human language. Pretraining LLMs on large corpora and fine-tuning them for specific tasks have yielded state-of-the-art results across various NLP applications, including summarization, question answering, sentiment analysis, and named entity recognition. Vaswani et al. [3] introduced the transformer architecture, which has been fundamental in summarization tasks due to its ability to handle long-range dependencies. Devlin et al. [1] introduced BERT, showcasing the effectiveness of masked language modeling for pretraining. Subsequent models like GPT-3 (Brown et al.) [4] further emphasized the scale's role in enhancing model capabilities. Howard and Ruder [5] proposed the Universal Language Model Fine-tuning (ULMFiT) method, demonstrating that LLMs could be fine-tuned with relatively small datasets to achieve excellent performance on downstream tasks. See et al. [6] discussed sequence-to-sequence models' challenges in producing coherent and concise summaries, highlighting the need for models that can generate novel sentences. BERT (Devlin et al. [1]) significantly advanced question answering tasks by enabling models to understand context better.

Later models, such as ALBERT (Lan et al., [7]), and T5 (Raffel et al., [8]), continued this trend, pushing the boundaries of model efficiency and performance. Models like REALM (Guu et al., [9]) integrate retrievable external knowledge bases to improve question answering performance, addressing knowledge gaps in pretrained corpora.

The adaptability of LLMs to sentiment analysis has been demonstrated through fine-tuning approaches, where models pretrained on general corpora are adjusted to understand nuances in sentiment (Sun et al., [10]). Expanding the scope of sentiment analysis, cross-lingual models (Conneau et al., [11]) and aspect-based approaches (Xu et al., 2019) offer detailed insights into sentiment across languages and specific aspects of products or services. Lample et al. [12] showed that LLMs could effectively capture the context necessary for identifying named entities in text, a foundational step for information extraction and knowledge graph construction. Akbik et al. [13] introduced Flair embeddings, emphasizing the importance of contextual representations for NER. These representations allow for the disambiguation of entities based on the surrounding text. While LLMs have shown remarkable success, challenges such as model interpretability, ethical considerations (Bender et al., [14]), and the environmental impact of training large models remain.

III. Model Description

The "Attention is all you need" paper proposes the model Transformer which is a novel architecture for sequence transduction tasks. It departs from traditional RNN-based models, instead relying solely on a multi-head self-attention mechanism. The Transformer's encoder-decoder structure utilizes masked self-attention in the encoder to capture long-range dependencies, while the decoder employs a combination of masked self-attention and encoder-decoder attention to focus on relevant parts of the source sequence during generation. Unlike previous architectures, there is no need of recurrence which allows for parallel processing and reportedly achieves state-of-the-art results in machine translation tasks with improved efficiency.

For selecting the model architecture, we first have to consider the different model architectures currently available. Many subsequent research efforts after the Transformer paper, simplified the architecture by removing either the encoder or decoder, opting for a single stack of either blocks. These blocks were stacked as high as computationally feasible and are trained on extensive text data, given that there is substantial computational resources to attain optimal performance.

For our task, we decided to choose a decoder-only based method, because they excel in few-shot learning scenarios by effectively leveraging the limited available data to generate accurate and coherent outputs. Their generative capability, attention mechanisms, and fine-tuning efficiency make them well-suited for tasks we have been asked to tackle. Furthermore, the decoder-only approach can be more robust than the encoder-only approach because it inherently contains more information about the input sequence through the attention mechanisms. In encoder-only models, the context about the input sequence is lost, making it challenging to generate meaningful output sequences. And lastly, choosing just the decoder-only based model will overall reduce the complexity of the model which would ideally lead to easier model interpretation, faster training times, and reduced computational overhead.

Given the context above, we implemented GPT-2 small from scratch, to tackle the tasks. GPT-2 (Generative Pre-trained Transformer 2) is a variant of the Transformer architecture designed and is decoder-only based.

- **Transformer Blocks:**
  GPT-2 small consists of 12 transformer blocks. Each block contains a self-attention mechanism with 12 attention heads. The dimensionality of the hidden layers in each transformer block is 768.
- **Embedding Layers:**
  GPT-2 small uses token embeddings to represent input tokens. The token embeddings have a dimensionality of 768, consistent with the hidden layer size of the transformer blocks. The positional embeddings used to capture the position of tokens in the input sequence also have a dimensionality of 768.
- **Layer Normalization and Residual Connections:**
  Each transformer block in GPT-2 small applies layer normalization after both the self-attention mechanism and the feedforward neural network. Additionally, residual connections are employed around each sub-layer (i.e., self-attention and feedforward network) in each transformer block.
- **Output Layer:**
  At the output of the model, a linear transformation followed by a softmax function is applied to generate the probability distribution over the vocabulary. The size of the vocabulary we still aren't sure of yet.

IV. Dataset

The pretraining corpus for the LLM is constructed using a carefully curated subset of the OpenWebText dataset ensuring a diverse and representative collection of text source. The Tiny Shakespeare corpus contains text data serves as the source for pretraining the language model, where the input is Text sequences, tokenized at the character level. Each character is encoded as an integer, and the sequences are batched for training. For each input sequence, the target output is the subsequent character sequence, shifted by one position, where the model learns to predict the next character given the current sequence.

SQuAD (Stanford Question Answering Dataset) dataset was used for the questioning/answering task. The input Context passages and questions are tokenized and encoded. These tokenized sequences are concatenated, and special tokens are inserted to delineate different parts of the input. The output is the answer text extracted from the context passage.

The CNN/DailyMail dataset is used for text summarization task. The input comprises the article text to be summarized. It is tokenized and encoded, with special tokens inserted to indicate the beginning and end of the input. The output is the human-generated summary corresponding to each article.

For the sentiment analysis, the input consists of text sentences extracted from the Financial Phrasebank dataset. Each sentence is tokenized and encoded into integer sequences using a tokenizer. Additionally, a special token is appended to denote the sentiment label associated with the sentence. The output corresponds to the sentiment label of each sentence. Sentiment labels are encoded as integers or strings, representing categories such as positive, negative, or neutral sentiment. The sentiment label is appended to the tokenized sequence as part of the output.

V. Implemented Experiments and Results

We trained the base model on the OpenWebText dataset using the GPT-2 architecture specified above. The loss curve can be seen in Fig 1.

For the fine-tuning tasks, we attempted two approaches:

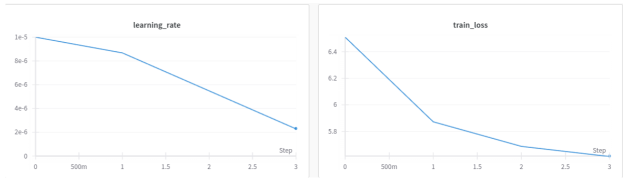

Fig. 1. Pretrained model loss curve

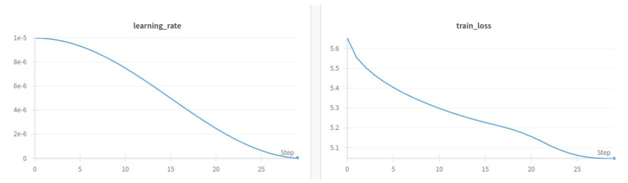

Fig. 2. Finetuned model loss curve

- Same model across all fine-tuning tasks by adding a task-specific prompt text. Therefore, all tasks are treated as text-generation tasks. The loss curve for the fine-tuned model can be seen in Fig 2
- Separate models for the text-generation and classification fine-tuning tasks

For all the text generation tasks, we used the same GPT-2 model specified in Section III, while for the classification texts, we created a new model with its backbone as the pre-trained GPT-2 model followed by a classification head.

In approach 1 in finetuning for the four tasks, we created task-specific processing of the datasets for each of them. Since GPT2 is a generative model and we have both generative and classification tasks, we decided to add queries between context and target output to inform the model what to generate next. Thus, in general, we format the input for the model as context + query + target output. For example, for the summarization task, we format the dataset as [‘article’] + “Here is the summary for the text:” + [‘highlights’]; and for sentiment analysis, we format the dataset as [‘sentence’] + “Here is the sentiment label for the sentence:” + [‘label’]. In this way, after training, when the model sees “Here is the summary for the text:” or “Here is the sentiment label for the sentence:”, it will know that the next word it should generate is about the summary of the model or the sentiment label instead of just human like language based on the context. Since we want the model to learn the four tasks at the same time, we shuffled all the data samples to create one dataloader for fine-tuning.

A. Summarization

To fine-tune for summarization, we added the prompt “Here is the summary for the text:”. We evaluated our performance using the BLEU score module available in PyTorch. We sliced the output string by removing the input text, the prompt, and everything after the ”—¡endoftext¿—” token in the text. We achieved a BLEU score of 96.7% on the test dataset.

B. Question Answering

To fine-tune for this task, we added the prompt ”Here is the answer for the question:”. We evaluated our performance using the ROUGE metric. This metric is used for evaluating automatic summarization and machine translation software in natural language processing. We chose to evaluate using ROUGE-N which denotes overlap of n-grams between the system and reference texts. We achieved a Rouge-2-P score of 0.006, Rouge-2-R score of 0.262, and Rouge-2-F score of 0.262. The performance is poor because our model generated a lot of text while the target is usually one word long. Changes in how we combine and batch the dataset might help with getting better ROUGE scores.

C. Sentiment Analysis

The LLM can be used for the sentiment analysis task by adding a task-specific prompt. With that objective, we fine-tuned our GPT-2 base model on the Financial Phrasebank dataset and obtained loss curve shown in Fig 2. However, because our base model isn’t very good and makes English errors, we could not evaluate this model against the dataset labels (positive, neutral, negative).

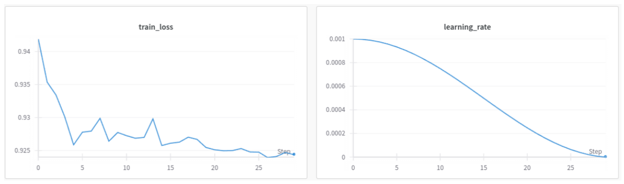

Fig. 3. Fine-tuning LLM model on sentiment analysis dataset using a classification head

So, we tried approach 2 using a classification head. The loss curve is shown in Fig. 3. This is a small dataset with 50257 samples. We created a train-test split of 88 : 12. We achieved an accuracy of 57.22% on the test set of 582 samples across 3 labels (positive, neutral, and negative).

D. Named Entity Recognition

We attempted to train a classification head based model for this task. However, since we our pre-trained model uses byte-pair encoding, and in this task we require the classification for each word, the classification-based model performed poorly.

VI. Conclusion

In this report, we employed a decoder-only architecture to train a Large Language Model and demonstrated its effectiveness when fine-tuned on four natural language processing tasks: summarization, question answering, sentiment analysis, and named entity recognition.

While our initial results are encouraging, there’s room for further improvement and exploration:

- Enhanced Pretraining: Expanding the pretraining corpus and exploring other pretraining methodologies could strengthen the LLM’s foundational language understanding.

- Refinement for NER: We encountered limitations when using the byte-pair encoding for the named entity recognition task. Exploring word-based tokenization methods would be interesting to tackle this problem.

References

[1] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[2] A. Radford, K. Narasimhan, T. Salimans, I. Sutskever *et al.*, "Improving language understanding by generative pre-training," 2018.

[3] A. Vaswani, N. Shazeer, N. Parmar *et al.*, "Attention is all you need," *arXiv preprint arXiv:1706.03762*, 2017.

[4] T. B. Brown, B. Mann, N. Ryder *et al.*, "Language models are few-shot learners," *arXiv preprint arXiv:2005.14165*, 2020.

[5] J. Howard and S. Ruder, "Universal language model fine-tuning for text classification," *arXiv preprint arXiv:1801.06146*, 2018.

[6] A. See, P. J. Liu, and C. D. Manning, "Get to the point: Summarization with pointer-generator networks," *arXiv preprint arXiv:1704.04368*, 2017.

[7] Z. Lan, M. Chen, S. Goodman *et al.*, "Albert: A lite bert for self-supervised learning of language representations," *arXiv preprint arXiv:1909.11942*, 2019.

[8] C. Raffel, N. Shazeer, A. Roberts *et al.*, "Exploring the limits of transfer learning with a unified text-to-text transformer," *arXiv preprint arXiv:1910.10683*, 2019.

[9] K. Guu, K. Lee, Z. Tung, P. Pasupat, and M.-W. Chang, "Realm: Retrieval-augmented language model pre-training," *arXiv preprint arXiv:2002.08909*, 2020.

[10] C. Sun, X. Qiu, Y. Xu, and X. Huang, "How to fine-tune bert for text classification?" *arXiv preprint arXiv:1905.05583*, 2019.

[11] A. Conneau, K. Khandelwal, N. Goyal, V. Chaudhary, G. Wenzek, F. Guzmán, E. Grave, M. Ott, L. Zettlemoyer, and V. Stoyanov, "Unsupervised cross-lingual representation learning at scale," *arXiv preprint arXiv:1911.02116*, 2020.

[12] G. Lample, M. Ballesteros, S. Subramanian, K. Kawakami, and C. Dyer, "Neural architectures for named entity recognition," *arXiv preprint arXiv:1603.01360*, 2016.

[13] A. Akbik, D. Blythe, and R. Vollgraf, "Contextual string embeddings for sequence labeling," *arXiv preprint arXiv:1808.03979*, 2018.

[14] E. M. Bender, T. Gebru, A. McMillan-Major, and S. Shmitchell, "On the dangers of stochastic parrots: Can language models be too big? ," *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency*, 2021.