Small LLM for SQuAD and CNN Tasks

Jeffrey Mi
jfmi@andrew.cmu.edu

Abstract

This paper presents the development and evaluation a decoder-only language model architecture with 67M parameters, tailored for question answering and summarization tasks. It was pretrained on the OpenWebText dataset. Subsequent fine-tuning was performed on the Stanford Question Answering Dataset (SQuAD) and the CNN/Daily Mail dataset to adapt the model for specific NLP tasks. Achieving a BLEU score of 0.492 on the CNN Daily Mail summarization task and 18.841 exact match on SQuAD QA, the results suggest the model requires further refinement to enhance its applicability in practical scenarios.

1 Introduction

Language models are becoming increasingly useful for real-world applications, however they often require large amount of compute to train and perform inference. Thus, it is useful to try and create usable models that are much smaller so that devices with limited compute can host them. In this paper, we will investigate the performance of a "decoder-only" small (67M param) model on a series of tasks. Our model's inputs will be a sequence of tokens, a numerical representation of sub-words. Our model will output the most likely next token in the sequence at each time-step in the sequence.

2 Literature review

Recent advancements in language modeling have been predominantly driven by developments in neural network architectures that emphasize efficient learning of word relationships. A pivotal innovation in this area was introduced by Vaswani et al. (2017) in their paper, "Attention is All You Need." The authors proposed the Transformer model, which discards traditional recurrence mechanisms in favor of self-attention mechanisms. This model efficiently computes global dependencies between input and output, facilitating significantly faster training times compared to its recurrent predecessors. The Transformer's ability to parallelize computations has set a new standard in the field, enabling the training of deep learning models on extensive text corpora more feasibly (Vaswani et al., 2017).

Building on the Transformer architecture, Radford et al. (2018) introduced the Generative Pre-trained Transformer (GPT) model, which further revolutionized language modeling. The GPT architecture uses a decoder-only Transformer architecture. It is initially pre-trained on a diverse dataset using unsupervised learning and later fine-tuned for specific tasks. This approach not only improved the quality of generated texts but also demonstrated impressive capabilities across a variety of language understanding benchmarks (Radford et al., 2018).

For researchers and practitioners looking to implement GPT-like models on a smaller scale, the minGPT and nonGPT repositories, written by Karpathy, serve as a great educational resource for building small to medium-sized language models.

37th Conference on Neural Information Processing Systems (NeurIPS 2023).

3 Model description

Our model uses a decoder-only transformer architecture with 5 decoder blocks. Tokens are first put through an embedding layer that outputs 512 dimension embedding vector for each token. They are next put through a positional encoding layer, then passed to the decoder. A decoder block has the following architecture:

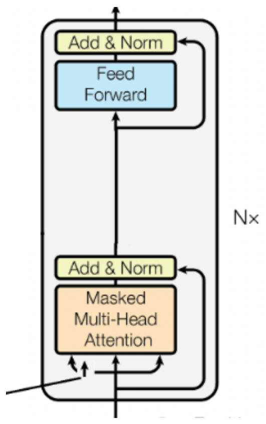

Figure 1: Decoder block

The decoder utilizes causal (masked) self-attention to prevent information flow from later time steps. After the decoder blocks, we have a final LayerNorm layer and a final linear layer that projects our 512 dimensional vectors to the vector space of our vocab size (50304).

The model has a total of 67M parameters. Table 1 highlights important model hyperparameters that were used for the final model, with d_{model} representing the token embedding vector size. Context length refers to the number of tokens that are passed into the model at once.

Table 1: Model hyperparameters

d_{model}	512
context length	512
num decoder blocks	5
attention heads	8
dropout	0.1

4 Dataset

4.1 OpenWebText

Our model was pretrained on the OpenWebText dataset, a dataset created by scraping web pages which were linked to by Reddit posts above a certain upvote threshold.

4.2 SQuAD for QA dataset

The Stanford Question Answering Dataset (SQuAD) is a benchmark for machine learning models on the question answering task. It includes over 150,000 questions on Wikipedia articles.

To make this task understandable for our language model we used a simple text formatting of the following format:

"Context: <context> Question: <question> Answer: <answer>"

4.3 CNN Daily Mail for Summarization Task

The CNN/Daily Mail dataset is a standard benchmark for evaluating natural language processing models on the summarization task. It consists of news articles from the CNN and Daily Mail websites paired with multi-sentence summaries. These summaries are highlights created by professional journalists, encapsulating key points of the articles.

To make this task understandable for our language model we used a simple text formatting of the following format:

"Article: <article> Summary: <summary>"

Many of the articles are longer than 512 tokens, so we simply truncate the articles so that the task embedding and article can fit within this context.

4.4 Tokenization

This model utilizes the GPT-2 tokenizer available through the tiktoken library. GPT-2 encodings have a total vocab length of 50304.

4.5 Batch sampling method

For pretraining, we concatonated all of the tokenized articles from the OpenWebText dataset, and randomly sampled batches of 32 512-token sequences to perform prediction. For fine tuning, since the CNN dataset is much larger than the SQuAD dataset, we reduced it to be the same size as the SQuAD dataset with respect to their number of articles. Then we concatonated these two tokenized dataset and trained our model on a shuffled dataloader on this dataset, as to not bias the fine-tuned model to either task.

5 Evaluation metrics

For the SQuAD QA task, we are simply evaluating our model's performance by getting the percentage of answers that were exactly the same as the given answers in the dev set. The SQuAD dataset comes with a evaluation script which computes this percentage by normalizing the predicted answers and comparing them to a number of viable answers.

For the summarization task, we use BLEU to compare our summarization with the reference. BLEU, or the Bilingual Evaluation Understudy score, quantifies the quality of machine-translated text by comparing it to high-quality reference translations. The calculation involves the following steps:

1. **N-gram comparison**: Counts the maximum number of times an n-gram (a sequence of n words) in the candidate translation appears in any reference translation. This count is clipped by the maximum count of that n-gram in the reference translations to avoid over-counting.
2. **Precision calculation**: For each n-gram length (typically 1 to 4), precision is calculated as the ratio of the clipped count of n-grams in the candidate that match those in the references to the total number of n-grams in the candidate.
3. **Breaching factor**: A penalty for too-short translations, known as brevity penalty (BP), is computed. If the length of the candidate translation is less than the reference, the BP penalizes the score:

$$\text{BP} = \begin{cases} 1 & \text{if } c > r \\ e^{(1-r/c)} & \text{if } c \leq r \end{cases}$$

   where c is the length of the candidate translation, and r is the effective reference corpus length.
4. **Score calculation**: The BLEU score is then calculated by taking the geometric mean of the precision scores for each n-gram length, multiplied by the brevity penalty:

$$\text{BLEU} = \text{BP} \cdot \exp\left(\sum_{n=1}^{N} w_n \log(p_n)\right)$$

where p_n is the precision of n-grams of length n, w_n are the weights assigned to each n-gram length (usually equal for all n-grams), and N is typically 4.

6 Loss function

To train our model, we are using cross-entropy loss. The formula for cross-entropy loss is given below.

$$L(y, z) = -\sum_{c=1}^{C} y_c \log(\text{Softmax}(z_c))$$

where y_c is a binary indicator (0 or 1) of whether class label c is the correct classification for the observation, and $\text{Softmax}(z_c)$ is the predicted probability of class c derived from the logits.

In the case of our language model, our logits would be the predicted distribution over all tokens for a given time step, and we would sum this over all time steps and batch dimension.

7 Results

A plot of our pretraining loss is shown below:

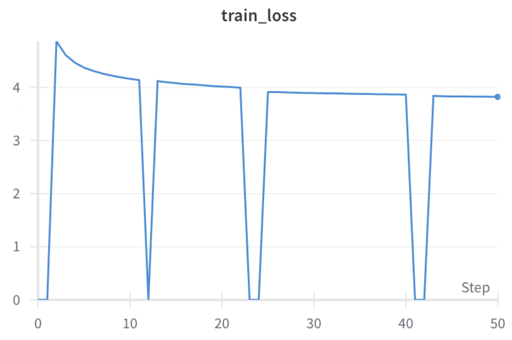

Figure 2: Training loss

Each iteration in Figure 1 represents an average loss over a 4000 batch window (0 loss iterations can be ignored). Our training CELoss plateaued around 3.8.

The results of our fine-tuning tasks can be seen in Table 2.

Table 2: Results

SQuAD (exact)	18.841
CNN Daily Mail (BLEU)	0.492

8 Discussion

The scores we achieve on the respective fine tuning tasks indicate that our small decoder-based model does not have much real-world application. Although it is small, and could more easily be hosted by devices with less compute power, it does not achieve good enough scores to justify its use in any real world scenario.

It should be noted that in order to achieve the best performance in the QA task, the output of the model needed to be parsed accordingly because of unreliable output. For example, in many of the answers, it prepended the string "Answer: " and did not in others, so the substring is removed if it is found in the answer. Additionally, the model outputs random variants of whitespace when giving answers which need to be stripped out.

9 Future works

In order to improve upon this model, different methods of pre-training and fine-tuning should be investigated. Perhaps different task-specific instructions would help the model make better predictions. Also, increasing context length could help since many of the articles from the CNN dataset were above 512 tokens. Experimentation with different pretraining datasets could possibly enhance performance on fine-tuning tasks as well.

10 Conclusion

Gauging our results from the fine-tuning tasks, our model does not perform well enough to be usable in any sort of real-life scenario. More experiments should be made with small language models to see if any real-world application can be found for their use.

References

[1] Hermann, K.M., Koçiskỳ, T., Grefenstette, E., Espeholt, L., Kay, W., Suleyman, M. & Blunsom, P. (2015) Teaching Machines to Read and Comprehend. In *Advances in Neural Information Processing Systems 28*, pp. 1693–1701. MIT Press.

[2] Karpathy, A. (2022) nanoGPT. GitHub repository. Available at: *https://github.com/karpathy/nanoGPT*.

[3] Radford, A., Narasimhan, K., Salimans, T. & Sutskever, I. (2018) Improving Language Understanding by Generative Pre-training. In *https://s3-us-west-2.amazonaws.com/openai-assets/research-covers/language-unsupervised/language_understanding_paper.pdf*.

[4] Rajpurkar, P., Zhang, J., Lopyrev, K. & Liang, P. (2016) SQuAD: 100,000+ Questions for Machine Comprehension of Text. In *arXiv preprint arXiv:1606.05250*.

[5] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., & Polosukhin, I. (2017) Attention is All You Need. In *Advances in Neural Information Processing Systems*, pp. 5998–6008.