CMPSC 497 Final

Eugene Kwek

Problem Definition

Tokenization enables the conversion of a large corpus of raw text into tokens that can be understood and processed by LLMs. Each token can represent a full word, a subword, or even individual characters. This helps compress raw text and make LLMs process language more efficiently and generalize better. However, tokenization also leads to problems. Because tokens do not correspond one-to-one with words or characters, modern LLMs struggle with reasoning about counting words/characters and spelling. This was famously brought to light last year with the Strawberry Problem, when state-of-the-art models such as GPT-4 were unable to determine the number of r's in 'strawberry'. This project focuses on these issues with using tokenization for large language models, and whether it is possible to improve LLM performance on character counting.

Dataset Construction

To create the dataset, we find a corpus of English words. Then, for each word, we form a training example with the following format:

Q: How many (character)'s are in (word)?
A: (word) has (number) (character)'s.

The character is chosen randomly from any character that is in the word. We use a freely available list of English words located on Github with 370,000 words total. We randomly sample 10,000 words to use as our training set, and another 100 for our test set.

Training

We trained Llama-3.2 1B on our dataset. This provided a good balance between model size and performance. We used Unsloth to train with a low-rank adapter (LoRA) with rank 64 and alpha 64. Because the model is relatively small, we use a high batch size of 128, with a learning rate of 10^{-3}. To improve performance, the model was only trained on responses and not prompts. Training was performed using a 2xT4 instance available on Kaggle.

Results

We have 100 examples in the test set, and the accuracy metric was used to benchmark the model's performance. We compared the original model with the new, fine-tuned model. The results are in the table below:

	Accuracy
Llama 3 1B Baseline	33%
Llama 3 1B Fine-tuned, 1 epoch	52%
Llama 3 1B Fine-tuned, 10 epochs	87%

Based on the results, fine-tuning the model on just one epoch dramatically improved the ability for Llama 3 to count characters in words, from correctly answering 33% of the time to 52%. As an experiment, we trained for 10 epochs as well. This could improve performance but introduced the risk of overfitting on the training set. However, training for longer appears to be a benefit, and our final accuracy was 87%. This is a >5x improvement in error rate over the base model. Therefore, we conclude that the shortcomings of using subword tokenization can be overcome with additional training.

Discussion

Throughout this project, I learned how to train LLMs for a research problem. I was quite surprised at my results, since I didn't expect such a large improvement with relatively few examples to train on. Still, there are some issues with the approach I took. All of my examples followed the same rigid chat format, which is not realistic for real-world use. Because of this, the model's general capabilities were likely harmed by the training run. A next step in this research would be to see if it is possible to train the model so that it improves at counting characters while also retaining its original chat performance.