ChatOPT (OpenWebText Pretrained Transformer)

Claudia Porto

Abstract

Since we coincidentally took a Natural Language Processing class right when ChatGPT was all the rage, it only made sense to do our final project on the finest neural network for language modeling around: the transformer. We also decided to give a nod to GPT-2, the predecessor of GPT-3, which is the foundation of ChatGPT, by fine-tuning and training it on the OpenWebText dataset, which contains web content from URLs shared on Reddit. Given these elaborate ideas, we had the basis for our project. We decided upon fine-tuning GPT-2 on this dataset to produce text generation output based on web article content. Throughout this project, we learned a lot about the hardships of data preprocessing, which must happen before the transformer can be trained, as well as how strenuous the machine learning process is on almost every single component of a computer (SSD, CPU, GPU, and RAM). Despite facing challenge after challenge, we were able to get our program running and our model trained exactly as we had intended.

1 Introduction

As mentioned earlier, the project's initial concept was to merge Project 1 and the in-class fine-tuning a pre-trained model activity. We aimed to create an original text generation project using sophisticated language models. To accomplish this, we planned to fine-tune GPT-2 on the OpenWebText corpus. The project's structure involved three major steps: preprocessing, training, and evaluation. During preprocessing, we loaded the dataset, tokenized it, and mapped it. Next, we defined the training specifications and parameters necessary before training the model. Finally, the evaluation stage provided insights into the model's performance. However, due to limited computing resources, this stage was challenging. The entire process was more demanding than we anticipated, but we learned valuable lessons. Despite its challenges, I'm pleased with the project's outcome, as it allowed me to enhance my understanding of language models, transformers, and the overall language model training process.

2 About Our Team

Our team of four included Hope Crisafi, Caleb L'Italien, Marielise Robillard, and myself. I took on the challenge of the coding work and problem-solving aspects of the project with the big help of Hope. Caleb was invaluable in refining the project idea, which started as a complex concept, to make it feasible within our given timeframe.

We utilized our expertise in Python, as well as the knowledge we gained in this class on transformers and fine-tuning models. Through the process of trial and error, we were able to truly grasp an understanding of the processes of fine-tuning a transformer model as well as all of the steps that go into data preprocessing such as tokenization and training.

3 Related Work/Inspiration

The idea was inspired by Project 1: Character-Based n-gram Language Models, where a character-based n-gram language model was trained on the works of William Shakespeare. By using the Shakespeare corpus as training data, the language model was able to generate a fixed length of text that resembled Shakespeare's writing style.

In this project, the goal was to build a language model that can take in a user prompt and generate text based on that input. The model used for this task was GPT-2, a powerful transformer-based language model.

To fine-tune the GPT-2 model, our team used techniques learned in class for fine-tuning the BERT model from the HuggingFace library on Google Colab. Specifically, we used the HuggingFace datasets library to load the OpenWebText

dataset. We then preprocessed the data by tokenizing the text using the GPT-2 tokenizer and splitting it into fixed-length sequences. Finally, we used the preprocessed data to fine-tune the GPT-2 model using the HuggingFace Trainer API. The model was trained to generate text that closely matches the style and content of web articles based on the user prompt. The resulting model can be used to generate new text content based on a given prompt.

4 OpenWebText Dataset

The OpenWebText dataset is a large corpus of web content scraped from URLs shared on Reddit. It was created as an open-source alternative to OpenAI's WebText dataset and contains text from web pages that received at least 3 karma points on Reddit, indicating community approval of the content quality.

The dataset contains diverse content covering a wide range of topics including technology, science, politics, entertainment, sports, and more. The text data has been preprocessed to remove various types of noise such as excessive whitespace and formatting issues.

The dataset is substantial in size, containing approximately 8 million documents with text extracted from web articles and pages. The dataset consists of plain text content with each entry representing the full text of a web page. Due to limited computing resources and the incredibly large nature of the dataset, we decided to use only 500 examples from the training data for our fine-tuning process. This smaller sample allowed us to complete the training within our resource constraints while still demonstrating the fine-tuning process and evaluating model performance.[1]

The OpenWebText dataset is designed for use in natural language processing (NLP) tasks such as text generation, language modeling, and more. It provides researchers with a large and diverse corpus of text data for training and evaluating NLP models.

The dataset is available for free download from the Hugging Face website, and it can be accessed and used with a wide range of popular NLP frameworks and tools, such as PyTorch, TensorFlow, and Transformers.

[1] https://huggingface.co/datasets/Skylion007/openwebtext

Text Content Examples
"WHAT?! DISNEY HAS A DONUT SUNDAE AND I DIDN'T KNOW ABOUT IT?! The Main Street Plaza Ice Cream Parlor can be found at the intersection of Main Street USA and Tomorrowland. The sundae comes with a warm glazed donut, warm apple compote, vanilla ice cream, chocolate sauce, whipped cream, a cherry, chocolate chips, and peanut butter chips. At press time, this costs $5.99..."
"Albert Einstein's theory of general relativity has been confirmed by observations of gravitational waves. The detection represents a milestone in physics and astronomy, offering a new way to observe the universe beyond light-based telescopes..."
"The new smartphone features a larger display, improved camera system with night mode, and extended battery life. Preorders begin next week with shipping expected in early March. Prices start at $799 for the base model..."
"Climate scientists report that global temperatures continue to rise, with the past decade being the warmest on record. The findings emphasize the urgent need for action to reduce greenhouse gas emissions and transition to renewable energy sources..."

Table 1: Text content examples from the HuggingFace library's OpenWebText dataset

5 GPT-2 Model

HuggingFace's GPT-2 model is a popular, large-scale transformer-based language model that has been pre-trained on a diverse corpus of internet text, comprising approximately 45 million links posted by Reddit users, called WebText. With 1.5 billion parameters, the model can be utilized for various natural language processing (NLP) tasks, including text generation, language translation, and sentiment analysis. The pre-training process was done using a self-supervised approach, where the model was trained on raw text data without any human labeling, enabling it to use publicly available data extensively.[2]

[2] https://huggingface.co/gpt2

During pre-training, the GPT-2 model learns to predict the next word in a sentence by generating inputs and labels from continuous text sequences of a specific length. The labels are shifted one token to the right, and the model uses an internal mask mechanism to ensure that predictions for each token only use input data up to that token and not future tokens.

By pre-training on massive amounts of English language data, the GPT-2 model learns a powerful internal representation of the language, which can be used for various downstream tasks. Although the model can perform various NLP tasks, it excels primarily at text generation from prompts, which is what it was primarily trained for.

The GPT-2 model is available through HuggingFace's transformers library, which is an open-source library providing access to various pre-trained language models. The transformers library also provides tools for fine-tuning pre-trained models on custom datasets, such as the OpenWebText dataset used in this project.

6 Data Pre-processing

To develop our model, we began with preprocessing, which entailed loading the dataset into our program, tokenizing it, and then mapping it.

Tokenization is a fundamental step in natural language processing (NLP) tasks such as language modeling, machine translation, sentiment analysis, and text generation. Once the dataset is tokenized, the resulting tokens can be fed into a model to perform various NLP tasks. In this project, we use the AutoTokenizer, the GPT-2 model, and the OpenWebText dataset from the HuggingFace library.

The Hugging Face AutoTokenizer is a tokenizer that automatically selects the appropriate tokenizer to use for a given language and model architecture.

When used with the GPT-2 model, the AutoTokenizer selects the GPT-2 tokenizer, which is specifically designed for use with the GPT-2 architecture. The GPT-2 tokenizer breaks down the input text from the OpenWebText dataset into a sequence of subword tokens, which are then fed into the GPT-2 model for further processing.[3]

The AutoTokenizer also handles additional processing steps such as adding special tokens for the beginning and end of sequences, padding sequences to a fixed length, and encoding the tokenized input in a format that the GPT-2 model can understand.

By using the AutoTokenizer, the process of selecting and configuring an appropriate tokenizer for a given model architecture is simplified, allowing us to focus on training and evaluating the model without worrying about the details of the tokenization process. The AutoTokenizer also supports a wide range of languages and model architectures, making it a versatile and powerful tool for natural language processing.

7 Training

Once we finished preprocessing the data, we could proceed to train our model. However, before doing so, we had to define certain training parameters. To accomplish this, we created a variable called "trainingArgs," which contained specifics regarding how our model would be trained. For instance, we opted to use the epoch strategy, which involves iterating through the entire dataset and processing batches of data while updating the model's weights. We also specified the learning rate, which we set to 2e-5. The learning rate determines how quickly the model adapts to the training data by controlling the step size of updates to the model's weights. In the interest of conserving our scarce computing resources and preserving GPT-2's existing learned data, we chose a relatively lower learning rate. Lastly, we set the weight decay to 0.01. This value helps prevent overfitting by encouraging the model to learn less complex weights. Our approach seemed to have worked since we achieved a training loss value ranging between 3.2-3.4 in each epoch.

8 Evaluation

The evaluation method on the Trainer object evaluates the performance of the trained model on a held-out validation dataset. In our code, the same language modeling dataset, lm_datasets, is used as both the training and evaluation dataset. We measured the model's effectiveness using the performance metrics commonly used for text generation, including Perplexity and Average Loss. Our evaluation performance metrics values are reported in Table 2.

Typically, to prevent overfitting to the training data, it is recommended to use a distinct validation dataset for evaluating the model during training. However, since we used a small subset of the Open-

[3] https://huggingface.co/docs/transformers/model_doc/auto

Epoch	Training Loss	Validation Loss
1	No log	3.263235
2	No log	3.236820
3	No log	3.227087

Table 2: Average Loss for Training and Validation Splits

Perplexity
25.21

Table 3: Perplexity for Validation Split

WebText dataset and did not have enough resources to create a split for both a train and validation set, we used it as both training and validation datasets.

9 Results/Future Work

As discussed in our Evaluation section, we had limited results to evaluate due to the scale of our project and limited computing resources. During the training process, we were able to monitor the validation loss, which decreased slightly with each epoch, indicating progress. We concluded that metrics like accuracy, precision, and F1 scores would not be suitable for evaluating our text generation model, so we opted to calculate the perplexity score, which resulted in 25.21, a relatively good value for a model trained on only 500 examples. However, we only used 500 examples from our dataset, so this may not accurately represent what the model could achieve with more training data.

We successfully implemented text generation capabilities with our fine-tuned model. The model was able to take user prompts and generate coherent text continuations based on the OpenWebText training data. While the generated text showed that the model learned patterns from the training data, the limited training set size (500 examples) meant the outputs were sometimes repetitive or less diverse than they might be with more training data.

In terms of future work, training on a larger subset of the OpenWebText dataset would likely improve the model's text generation quality and diversity. With more computing resources, we could use thousands or tens of thousands of examples instead of just 500, which would give the model more patterns to learn from and likely result in more varied and interesting generated text. Additionally, implementing more sophisticated evaluation metrics beyond perplexity, such as human evaluation of text quality or diversity metrics, would provide better insight into the model's true capabilities.

10 Conclusion

While we knew that Natural Language Processing projects can be challenging, we did not anticipate encountering issues with nearly every line of code we executed. Despite these difficulties, however, we found that each problem we solved helped us to understand the project on a deeper level. By breaking down each subsystem and identifying the best solution for each issue, we gained a comprehensive understanding of the entire process. We believe that the most effective way to learn programming is to tackle real-world projects and work through the problems that arise. The more issues you encounter and resolve, the more you learn and grow as a programmer. Ultimately, despite facing challenges related to the size of the model and computing resources, we were able to achieve our objective of fine-tuning the model on the OpenWebText dataset and obtaining evaluation data. We successfully trained the model on 500 examples, achieved a perplexity score of 25.21, and demonstrated text generation capabilities with our fine-tuned model. We are pleased with the final outcome and plan to revisit this project in the future with the aim of training on larger subsets of the data to improve the model's text generation quality and diversity.

11 Our Experience

As previously mentioned, this project presented a multitude of challenges, and at nearly every step of the process, we encountered difficulties. We were able to make significant progress, despite slow training and loading times, and I have surely gained a newfound appreciation for those who work on projects of this nature regularly. This field of Computer Science and AI is among the most challenging, and I now understand why.

The sheer scale of computing resources required for a project of this magnitude was severely underestimated on my part. I am grateful for those who have already trained some of the most widely used language models today, as I can only imagine the exorbitant cost of a room filled with 14,000 GPUs.

Despite the challenges we faced, this project has deepened my understanding of the entire training process. We encountered issues early on with dataset compatibility, as the original Reddit dataset

we planned to use was no longer supported by HuggingFace. This required us to pivot to the OpenWebText dataset and adjust our project scope. We also faced challenges with tokenization and ensuring the model could properly process the input sequences. Then, we encountered issues with tensor creation, as the sizes of the tensors were not uniform. These issues needed to be resolved before we could train our model effectively. Due to limited computing resources and disk space, we made the pragmatic decision to train on only 500 examples rather than the larger dataset we initially planned.

Our efforts to obtain evaluation metrics were also met with challenges. We successfully calculated the perplexity score and implemented text generation with our fine-tuned model. Through careful resource management and problem-solving, we were able to complete the training process and generate text output, demonstrating that even with limited resources, meaningful NLP projects are achievable.

Honor Code

I affirm that I have carried out my academic endeavors with full academic honesty.

Claudia Porto

Acknowlegments

The Natural Language Processing queen: Professor Kristina Striegnitz.