CS 449 Final Project Proposal

Due: October 17, 2024, at 11:59 pm

1. Names

Qingyuan (Jason) Yao, Murat Sankaya

Please also put each group member’s name and email in the README.md of your repository.

2. Logistics

We will work every weekend for 4 hours each week, specifically on Saturday from 1 to 5 p.m. Communication will be via WhatsApp.

3. Abstract

Our final project aims to recreate the seminal "Attention is All You Need" paper on transformers using PyTorch. We plan to implement the model and evaluate it with a range of tests, spanning from simple to more complex tasks.

4. Introduction

This project holds significant importance to both of us, as we have prior experience working with transformer-based models and are eager to understand their underlying mechanics more deeply. Although we have previously attempted to study the "Attention is All You Need" paper, we struggled to fully grasp its concepts. Through this project, we aim to gain a much deeper understanding of how transformers function and develop a strong intuition about the model’s inner workings.

We will be working with sequenced text data, which we will describe in more detail in Section 5. Our focus will be on evaluating the attention mechanism and comparing its speed of execution against more complex recurrent and convolutional networks that also incorporate attention mechanisms.

5. Data

5.1 Describe your dataset(s)

For our project, we plan to use a range of datasets, starting with synthetic toy datasets and progressively increasing complexity.

1. Arithmetic Operations Dataset (Synthetic)

- **Description**: This is a simple, synthetic dataset where the task is to predict the result of basic arithmetic operations like addition or subtraction. The input sequence is an arithmetic expression such as "2 + 3", and the output is the result of the operation, in this case, "5".
- **Why**: This dataset allows us to test whether the transformer can handle basic sequence-to-sequence mappings.

- **Data Representation**: Sequences of arithmetic expressions with varying lengths. Each sequence consists of digits, operators, and spaces.
- **Example**:

```
# Example data
data = [("2 + 3", "5"), ("5 - 2", "3"), ("7 + 8", "15"), ("9 - 6",
"3")]
```

- **Complexity**: Low.

2. Reverse a String Dataset (Synthetic)

- **Description**: In this dataset, the task is to reverse a given string. For example, the input "hello" would result in "olleh". This helps test the model's ability to learn positional dependencies.
- **Why**: Slightly more complex, this task tests the ability of the transformer to understand the structure of a sequence.
- **Data Representation**: Strings of varying lengths represented as sequences of characters.
- **Example**:

```
# Example data
data = [("hello", "olleh"), ("world", "dlrow"), ("transformer",
"remrofsnart")]
```

- **Complexity**: Medium.

3. WikiText-2 (Decoder model test)

- **Description**: The WikiText dataset contains full Wikipedia articles. WikiText-2 is a smaller version with about 2 million tokens, The task is to predict the next word in a sequence (language modeling).
- **Why**: This dataset helps evaluate how well the transformer handles long-range dependencies and performs in generating coherent text sequences.
- **Data Representation**: Sequences of words representing full Wikipedia articles, with sequence lengths often exceeding 500 words.
- **Source**: WikiText-2 Dataset
- **Complexity**: High.

4. GLUE Benchmark (General Language Understanding Evaluation) (Encoder test)

- **Description**:A collection of diverse natural language understanding tasks, including sentiment analysis, paraphrase detection, and more. It provides a wide variety of tasks for testing language model capabilities.
- **Source**: GLUE Benchmark
- **Complexity**: High

5. WMT 2014 English-German Translation Dataset

- **Description**: This dataset is widely used in machine translation tasks and is provided by the Workshop on Machine Translation (WMT). It has been frequently used to benchmark models, including the original Transformer model from the "Attention is All You Need" paper.
- **Content**: The dataset contains around 4.5 million sentence pairs of English and German. The sentences range from short phrases to longer, more complex sentence structures, making it ideal for training and evaluating sequence-to-sequence models like transformers.
- **Data Representation**: Each sentence in the dataset is tokenized into a sequence of words or subwords using techniques like Byte-Pair Encoding (BPE). Sentence length can vary, but the average sequence length is approximately 20 tokens.
- **Annotation and Labels**: The dataset is labeled in the sense that each English sentence has a corresponding German sentence, forming supervised pairs for training machine translation models.
- **Source**: WMT 2014 English-German Translation Dataset

We will add more dataset to test as we progress.

5.2 Load your dataset(s)

Demonstrate that you have made at least some progress with getting your dataset ready to use. Load at least a few examples and visualize them as best you can.

```
import nltk
from transformers import GPT2TokenizerFast
from nltk.tokenize import word_tokenize

# Download NLTK punkt tokenizer if not already present
nltk.download('punkt')

# Initialize GPT-2 tokenizer
tokenizer_gpt = GPT2TokenizerFast.from_pretrained("openai-community/gpt2")

# File paths for the dataset
file_names = ["wiki2.test.txt", "wiki2.train.txt", "wiki2.valid.txt"]

# Initialize lists to hold processed data
dataset = []
tokenized_data = []
gpt_tokens = []
gpt_tokens_ids = []

# Loop over the file paths
for path in file_names:
    file_path = "/content/drive/My Drive/CS449Group/" + path
    with open(file_path, 'r', encoding='utf-8') as file:
        text = file.read()

        # Tokenize using NLTK
        nltk_tokens = word_tokenize(text)

        # Tokenize using GPT-2
        gpt_token_ids = tokenizer_gpt(text)["input_ids"]
        # If you want to chunk the GPT token IDs for processing in a
transformer model
        max_length = 1024
        gpt_chunks = [gpt_token_ids[i:i + max_length] for i in range(0,
len(gpt_token_ids), max_length)]

        # Convert GPT-2 token IDs back to tokens for each chunk
        for chunk in gpt_chunks:
            tokens = tokenizer_gpt.convert_ids_to_tokens(chunk)
            gpt_tokens.append(tokens)

        # Append the data to corresponding lists
        gpt_tokens_ids.append(gpt_token_ids)  # Store full token IDs
        dataset.append(text)  # Store original text
        tokenized_data.append(nltk_tokens)  # Store NLTK tokens

# After processing, you can print or visualize the results:
print(f"Number of files processed: {len(file_names)}")
print(f"First few GPT tokens from the first file: {gpt_tokens[0][:20]}")
print(f"First few NLTK tokens from the first file: {tokenized_data[0]
[:20]}")

Result:
Number of files processed: 3
First few GPT tokens from the first file: ['Ġ', 'Ċ', 'Ġ=', 'ĠRobert',
'Ġ<', 'unk', '>', 'Ġ=', 'Ġ', 'Ċ', 'Ġ', 'Ċ', 'ĠRobert', 'Ġ<', 'unk', '>',
'Ġis', 'Ġan', 'ĠEnglish', 'Ġfilm']
First few NLTK tokens from the first file: ['=', 'Robert', '<', 'unk',
'>', '=', 'Robert', '<', 'unk', '>', 'is', 'an', 'English', 'film', ',',
'television', 'and', 'theatre', 'actor', '.']
```

5.3 Small dataset

Many deep learning datasets are very large, which is helpful for training powerful models but makes debugging difficult. For your update, you will need to construct a small version of your dataset that contains 200-1000 examples and is less than 10MB. If you are working with images, video, or audio, you may need to downsample your data. If you are working with text, you may need to truncate or otherwise preprocess your data.

Give a specific plan for how you will create a small version of one dataset you'll use that is less than 10MB in size. Mention the current size of your dataset and how many examples it has and how those numbers inform your plan.

Plan to Create a Small Dataset (<10MB):

To create a small, manageable version of the WikiText-2 dataset or other large datasets that fits within the size limit of **less than 10MB**, we will:

1. **Subset the Dataset**: - We will sample around **500–1000 sentences** from the original dataset. Based on average token sizes, this should give us a dataset under 10MB. - To ensure variety, we will select samples from both the **train** and **test** sets.

2. **Truncate Long Sequences**: - Wikipedia articles can be quite lengthy, so for this small dataset, we will limit the length of each sentence or sequence to **100 tokens** (words or subwords). This will ensure that we don’t have any overly long examples that take up too much space.
3. **Tokenization**: - We will use the **GPT-2 tokenizer** to tokenize the text into subword units. Subword tokenization allows us to compactly represent the text, while still capturing meaningful parts of the language, resulting in fewer tokens per sentence and thus reducing the dataset size.
4. **Verify Size**: - After tokenizing and truncating the data, we will measure the total size of the small dataset to ensure it is below 10MB. If necessary, we will further downsample by removing additional sentences.

6. Methods

Describe what methods you plan to use. This is a deep learning class, so you should use deep learning methods. Cite at least one or two relevant papers. What model architectures or pretrained models will you use? What loss function(s) will you use and why? How will you evaluate or visualize your model’s performance?

Our approach is straightforward: we will implement and test a transformer model (Vaswani et al., 2017). The initial plan is to evaluate our model on sequenced text data. However, we aim to maintain flexibility throughout the project to explore new ideas that may emerge. For instance, after developing the model, we may experiment with architectural variations. We might create an encoder-only architecture like BERT (Devlin et al., 2018) and apply it to a classification task, or a decoder-only model like GPT-2 (Radford et al., 2019) and evaluate it on simple text generation tasks.

The choice of loss function will depend on the specific task we are evaluating. If we replicate the experiments from the original paper, we will use **cross-entropy loss**, which is appropriate for sequence-to-sequence tasks. This loss will help us measure the difference between the model's predicted token distribution and the target token.

In cases where we recreate tests from Vaswani et al.'s paper, at least one of which we plan to use, we will evaluate model performance using the **BLEU score** (Papineni et al., 2002), a standard metric for machine translation. We will then compare our model’s performance to that of a complex pre-built recurrent neural network. To visualize the results, we will generate line charts showing the accuracy on the training and test sets as a function of the number of training steps for each model.

7. Deliverables

Include at least six goals that you would like to focus on over the course of the quarter. These should be nontrivial, but you should have at least one and hopefully both of your “Essential” goals done by the project update. Your “Stretch” goals should be ambitious enough such that completing one is doable, but completing both this quarter is unlikely.

7.1 Essential Goals

- **EG-Goal 1:** Develop the code for transformer model introduced in Vaswani et al. (2017)
- **EG-Goal 2:** Test the transformer model on at least one nontrivial(high complexity) dataset.

7.2 Desired Goals

- **DG-Goal 1:** Submit a well-documented Jupyter notebook that could serve as a tutorial for someone with limited knowledge.
- **DG-Goal 2:** Include code for visualizations and demos in the Jupyter notebook, such as visualizing how the attention mechanism works on a sequence of text.

7.3 Stretch Goals

- **SG-Goal 1:** Extend the code to different architectures.
  - **SG-Goal 1.1:** Implement a decoder-only version (Radford et al., 2019) and test it on a relevant dataset, the **WikiText-2**.
  - **SG-Goal 1.2:** Implement an encoder-only version (Devlin et al., 2018) and test it on a relevant dataset, the **GLUE Benchmark**
  - **SG-Goal 1.3:** Explore the application of the transformer model on different types of data beyond sequenced text, if time permits.

8. Hopes and Concerns

What are you most excited about with this project? What parts, if any, are you nervous about?

We are most excited about the opportunity to gain a deep understanding and intuition of the transformer architecture.

However, we are concerned about our limited familiarity with PyTorch, which may take time to learn. Additionally, we are eager to complete the implementation of the paper and move on to applying it to new problems, but we worry that the implementation process might take longer than expected.

9. References

Cite the papers or sources that you used to discover your datasets and/or models, if you didn't include the citation above.

> Papineni, K., Roukos, S., Ward, T., & Zhu, W. J. (2002). BLEU: A method for automatic evaluation of machine translation. *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, 311-318.
>
> Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., & Sutskever, I. (2019). Language models are unsupervised multitask learners. *OpenAI Blog*, 1(8).
>
> Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2018). BERT: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.
>
> Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). Attention is all you need. *Advances in Neural Information Processing Systems*, 30, 5998-6008.