lab2

group9	姓名
Team Member 1	贾世安(JIA SHIAN)
Team Member 2	陶毅诚(TAO YICHENG)

question1

Explain why transformers train(speculate i think) using entire sentences instead of short "lookback" sentences like LSTM.
(in fact, we use "batch_size" for training except "lookback")

- Transformers rely on self-attention mechanisms, where each word in a sentence attends to all other words within that sentence. This mechanism allows the model to capture the contextual relationships between all words simultaneously, regardless of their positional distance. In contrast, LSTM models process sequences sequentially, which restricts their ability to capture long-range dependencies effectively.
- Transformers can process sentences in parallel, making them highly efficient for training on modern hardware accelerators like GPUs and TPUs. Each word in a sentence can be processed independently, allowing for parallel matrix multiplications and speeding up training. This parallelism is not possible with LSTM models, as they rely on sequential processing and can only process one word at a time.
- Transformers use positional encodings to incorporate positional information into the model. This enables the model to learn the relative positions of words within the sentence. As a result, the model can handle sentences of varying lengths without losing positional information. LSTM models, on the other hand, do not inherently capture positional information and require additional mechanisms like padding or truncation to handle variable-length sequences.
- Transformers are particularly effective at capturing long-term dependencies in text. The self-attention mechanism allows information to propagate directly across all positions in the input, enabling the model to consider distant context when making predictions. LSTM models, although capable of capturing some long-term dependencies, can struggle with maintaining and propagating information over longer sequences due to the vanishing or exploding gradient problem.

question2

Explain why the sentences used to train the transformers must be of fixed length.

- **Batch Processing**: Transformers are commonly trained on mini-batches of sentences for efficient parallel processing. To form a batch, all input sequences within the batch must have the same length. By fixing the length of sentences, it allows the model to process multiple sentences simultaneously in a batch, which can significantly speed up training by leveraging parallel computing.
- **Memory Efficiency**: Transformers require a fixed-size attention matrix for each token in the sequence. The attention matrix captures the relationships between tokens. By using fixed-length sentences, the model can allocate memory efficiently for the attention matrices, regardless of the actual sentence length. This enables more efficient memory usage during training and inference.
- **Optimization Stability**: During training, the model's parameters are updated based on gradients computed through backpropagation. When using variable-length sequences, the gradients computed for different sentence lengths can vary, leading to unstable optimization. By fixing the sentence length, the gradients remain consistent across examples, which promotes stable and consistent updates to the model's parameters.
- **Hardware and Software Constraints**: Many deep learning frameworks and hardware accelerators have limitations on variable-length sequences. Using fixed-length sentences simplifies the implementation and utilization of these frameworks and accelerators, making training and inference more efficient and compatible with the available infrastructure.

question3

Using the Hugging Face website or otherwise, explain the parameters in AutoConfig.from_pretrained that we have used.

1. **model_name**: This parameter specifies the name or identifier of the pre-trained model. It is used to retrieve the corresponding configuration file from the Hugging Face model hub.
2. **vocab_size**: The vocab_size parameter is used to specify the size of the vocabulary for the model. It should match the number of unique tokens in the tokenizer associated with the model.
3. **n_ctx**: The n_ctx parameter sets the maximum length of the input sequences for the model. It determines the size of the attention matrices and positional encodings.
4. **bos_token_id**: The bos_token_id parameter specifies the token ID for the beginning of sentence (BOS) token. It represents the start of a sequence and is used for tasks like text generation.
5. **eos_token_id**: The eos_token_id parameter specifies the token ID for the end of sentence (EOS) token. It represents the end of a sequence and is used for tasks like text generation

question4

Compare the texts generated from the transformer that was trained from scratch versus the transformer that used the pretrained GPT2 weights. Do you see a difference in quality, e.g. fewer "non-English" words?

1. the texts generated from the transformer that was trained used the pretrained GPT2 has fewer "non-English" words for example, the thransformer that was trained from scratch will outputs something like "know't", ""."" , which is "non-English", but the pretrained is better.
2. pretrained transformer has less grammar errors