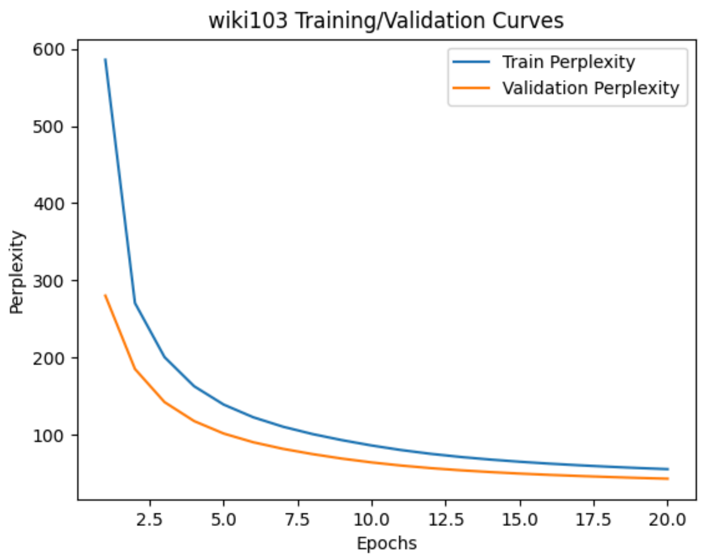
Team 2
Jack Hanke, Daniel Plotkin, Nicole Birova
NLP Homework 2
10 May 2025

Homework 2 Report

We follow the outlined approach for Question 1. For the data, we encode each question in the *fact-stem-option* format, and append the *[CLS]* and *[END]* tokens to the beginning and end of each sentence respectively. After tokenizing the four sentences and running the token streams through BERT, we multiply with a single linear layer to map the final hidden state to logits corresponding with the 4 possible answers A, B, C, and D. We will fine-tune using cross entropy for the true answers to the question. Before fine-tuning, we obtain a baseline test accuracy of 29.8%. After fine-tuning for 1 epoch with the Adam optimizer and a batch size of 5, we achieve validation and test accuracies of 61.8% and 64.2% respectively. Further improvements may be made by adding more layers between BERT inference and final output logits, which would correspond with adding more parameters for QA specification.

For 2, we first report our pretraining learning curve on the wiki103 dataset.



After 20 epochs of training, we achieve a final test perplexity of 42.03. To fine-tune this model on the QA dataset, we use the following prompt format:

<space> + <fact> + <space> + <question stem> + '[A] ' + <choice for A> + … + ' Answer: '

We directly train on the sequences created by this prompt, with the final answer appended to the shifted target sequence. We tried to use cross entropy loss over the entire sequence, ignoring the inserted padding token of 0. This did not work well for us. We saw significant improvement by just fine tuning on predicting the last token in the sequence.

Our baseline was 0.0% accuracy on the test dataset. After 5 epochs, we achieve a final validation and test accuracy of 40.0% and 40.0% accuracy. An example of our model predicting the correct letter is shown below.

Example prompt, for which the correct answer is *[B]*:

using less resources usually causes money to be saved A person wants to start saving money so that they can afford a nice vacation at the end of the year. After looking over their budget and expenses, they decide the best way to save money is to [A] make more phone calls [B] quit eating lunch out [C] buy less with monopoly money [D] have lunch with friends Answer:

Model prediction token index: *347*
Model prediction detokenized: *B*

The main limitations for this approach is the substantial pretraining required to fine-tune a model of sufficient capability on such a task. Additionally, the larger vocabulary size hinders performance. As expected, the BERT classifier performs better, as the model only predicts over the 4 possible letters, while the generative response classifies over 50k+ tokens.

For 3, we implement a basic greedy decoder, where we generate new tokens until the end-of-sequence token is generated or 20 tokens are generated. Though this code does autoregressively generate tokens, we were unable to demonstrate any meaningful generation, even after considerable effort. Using the same example sentence, after finetuning on the full sequence, we generate:

Example prompt, for which the correct answer is *[B]*:

using less resources usually causes money to be saved A person wants to start saving money so that they can afford a nice vacation at the end of the year. After looking over their budget and expenses, they decide the best way to save money is to [A] make more phone calls [B] quit eating lunch out [C] buy less with monopoly money [D] have lunch with friends Answer:

Model prediction:

's food to get more money to get more money to get more money [D] to get more

Clearly nonsense, and it seems to pick the last letter just to be wrong on top of being nonsensical. Of all of our testing, the “best” generation we achieved was on the following example prompt.

Example sentence:

what is a tomato?

Model prediction:

[A] a plant [B] a plant's plant [C] a plant's plant's

At least *[A]* is a real answer. Clearly the model is beginning to understand the styling of the QA dataset, but its ability to predict the answers was so weak we were unable to demonstrate anything resembling what this question might hope. The performance metrics, though calculated, are expectedly abysmal.

	BLEU	**ROUGE (1)**	**BERTscore (F1)**
Validation	0.0000	0.0000	0.0000
Test	0.0000	0.0001	0.0000

For validation and test accuracy, we use masked accuracy over the entire response, and achieve 1.11% and 6.25%. This likely is the result of learning to copy words seen, and nothing resembling real understanding.

Strangely, including the end-of-sequence token in training led the model to just predict this token, resulting in no generation. Additionally, fine tuning on more than one epoch resulted in even worse generation (what is even worse? random punctuation, mostly semicolons).

We were disappointed in the results. As expected, purely generative QA was tricky, but our team did not expect how much so.