Report

My solution uses the dslim/bert-large-NER model and tokenizer. The training process consisted of:

- Tokenizing the texts: the texts in my dataset were already tokenized into words, but I had to convert the given word tokens to subword tokens feedable to the model. I also had to adjust the labels accordingly.
- Metrics: I used F1 Score, due to the imbalance in the data (clearly, mountain names are much more rare than other words).
- Class weights: I created a custom trainer that balanced class weights to account for class imbalance in the data.
- Training itself: the model was trained for 5 epochs with a learning rate of 2e-5.
- Evaluation: the model achieves F1 Score of 87.42% on test data.

I see several potential ways to improve my model:

- Hyperparameter tuning: due to time limits, I did not do much research in this direction. I played a little with learning rate, but there is a lot more to do here, e.g., use learning rate decay. I could try training for more epochs; given the dynamics of F1 Score during the first 5 epochs, this does not look like a prospective direction, but I could still study this a little more, starting with plotting the loss curve and thinking based on that. There are more hyperparameters, like weight decay.
- Different model architecture: I tried several other BERT models, like elastic/distilbert-base-cased-finetuned-conll03-english and dslim/bert-base-NER. Among those, the one I used worked best with my hyperparameters. Still, I have seen many more on HuggingFace, and I could try to use entirely different architectures, like spacy, which is a popular framework for NER.
- In the demo I viewed several examples of texts which successfully deceived my model; I could delve deeper in this direction, identifying mistakes and complementing the dataset with relevant samples.
- Data augmentation: the dataset is not exactly small, but not so big as not to want for augmentation.