CS689: Computational Linguistics for Indian Languages

Tushar Dupga, 231110053

March 13, 2024

Question: 5

Both models demonstrate strong performance in classifying NER tags. Indic-BERT achieved a macro-F1 score of 0.78, indicating its ability to identify and label different named entities within the text accurately. IndicNER performed even better with a macro-F1 score of 0.83, showcasing its slightly superior effectiveness in this task. However, compared to ChatGPT (given extensive context), the responses from IndicBERT and IndicNER sometimes required repeated generation to achieve similarly comprehensive answers.
The absence of two MISC tags in both models likely impacts their performance; incorporating these tags could potentially improve accuracy. Despite this limitation, both models performed well compared to manually classified data, achieving a commendable F1 score of around 0.85.

The hyper-parameters of the model tuned, along with their significance:

- **Number of Epochs :**The number of epochs (4) determines how often the model processes the entire training dataset. A value of 4 was chosen to prevent overfitting (indicated by increasing validation loss) while still allowing the model to converge sufficiently with a dataset of 40,000 sentences. Higher epoch counts might lead the model to memorize the training data too closely, while fewer could prevent it from learning the underlying patterns effectively.
- **per device train batch size :** The per-device train batch size (24) controls how many training samples the model processes in a single update step. This choice was a compromise between training speed and memory constraints. Smaller batch sizes can lead to slower training and more noisy updates, while larger batch sizes, although computationally efficient, risk out-of-memory errors on the available hardware.
- **per device eval batch size :** It is defined as the number of training samples provided to the model for prediction. I have chosen 16 as the batch size for the evaluation pass. This choice was a compromise between training speed and memory constraints. Smaller batch sizes can lead to slower training and more noisy updates, while larger batch sizes, although computationally efficient, risk out-of-memory errors on the available hardware.
- **evaluation strategy and save strategy:** I employed a step-based evaluation strategy to monitor the model's performance during training. After every 500 steps (mini-batch updates processed by the model), the model was evaluated on a separate validation dataset. This validation process helps assess how well the model generalizes to unseen data and prevents overfitting on the training data. During evaluation, the model's performance was measured using relevant metrics (not mentioned here, but could be F1 score or accuracy). The two models with the highest performance on the validation set were then saved as checkpoints. This checkpointing strategy allows you to revert to the best-performing model encountered during training if needed, even if training continues beyond that point.
- **weight_decay:** Weight decay combats overfitting by adding a penalty term to the model's loss function. This term is proportional to the square of the weight values, meaning larger weights are more heavily penalized. A value of 0.01 strikes a balance between regularizing overly complex models and still allowing the model to learn important patterns from the data.

The output of the both the Models:

For IndicBERT Fine-Tuned Model:

```
from transformers import AutoModelForTokenClassification, AutoConfig, AutoTokenizer, TrainingArguments,
import numpy as np

config = AutoConfig.from_pretrained('ai4bharat/indic-bert', num_labels=7, finetuning_task='ner')
tokenizer = AutoTokenizer.from_pretrained("ai4bharat/indic-bert")
model = AutoModelForTokenClassification.from_pretrained('ai4bharat/indic-bert', num_labels=7 )
```

Figure 1: IndicBERT Model

```
s = "अभिनेत्री सोहा अली खान से उनकी मां अभिनेत्री शर्मिला टैगोर खासी नाराज़ हैं."
print(get_predictions(s, tokenizerB, model_trained_Bert))
✓ 0.0s
['O', 'B-PER', 'I-PER', 'I-PER', 'O', 'O', 'O', 'O', 'B-PER', 'I-PER', 'O', 'O', 'O']
```

Figure 2: IndicBERT test

For IndicNER Fine-Tuned Model:

Figure 3: IndicNER Model

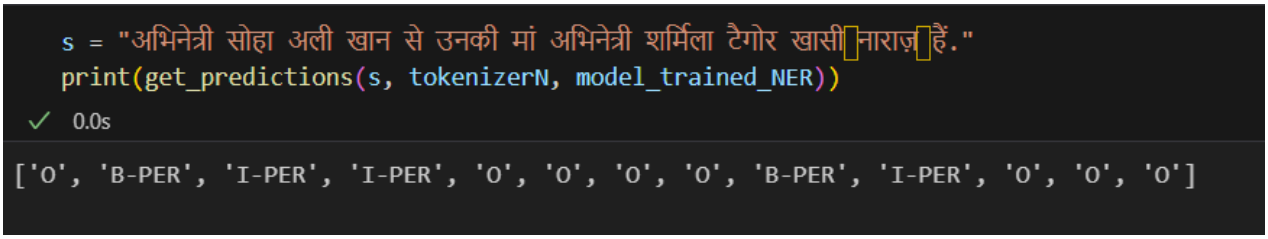

Figure 4: IndicNER test