CS689: Computational Linguistics for Indian Languages

Prashant Kumar, 231110036

March 12, 2024

Question: 5

The results in comparison indicate that both the Models are doing decent classification of **NER** tags. Although results against the comparison of OpenAI's ChatGPT prediction depend a lot on the context given to ChatGPT before the classification of NER (I have tried some and finally after the example given in the assignment is passed to ChatGPT, it classified better than before). The comparison indicates that ChatGPT prediction depends on context, and both models are doing well. Still, **it can be improved by more training and adding a class in the training data i.e., MISC** because no MISC label is present in the training data. The fine-tuned Models in question 2 have a macro f1-score of approximately 0.80 on the test data; similarly, the models achieve a similar macro f1-score against the manually classified data in question 1.

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

num_train_epochs: This hyperparameter represents the number of times the whole training dataset is passed in the model (one pass is when data is passed in both directions). Here, I have chosen 5 epochs that are good for updating weights to converge the model and also avoiding overfitting for the data of only 50000 training samples.

per_device_train_batch_size and per_device_eval_batch_size: Batch size represent the number of training samples that are being fed into the network at once for prediction. These parameters are the batch size for the train pass and for the eval pass respectively. I have chosen 32 as the batch size for the training pass and 16 for the validation pass. These values contribute to faster training and optimal use of available GPU for training, as a low value corresponds to slow training and a higher value leads to an out-of-memory error.

warmup_steps: This hyperparameter represents the number of times the gradient should increase linearly, and this helps the model to train optimally and not fall into local minima. I have chosen 500 steps as a warm-up for the gradient, which helps to not get stuck into local minima in the early training phase; later after 500 steps, the gradient will have the maximum value, and the model will continue on that.

weight_decay: The Weight decay parameter is used to avoid the overfitting of the model by regularizing the large weights. This parameter is selected as 0.01, which is a common value because it gives good penalization to large values and affects less to lower values.

evaluation_strategy and save_strategy: This is not a hyperparameter to tune, but it is used to select the evaluation pass at the end of each epoch, and the save_strategy corresponds to saving the model checkpoints after each epoch.

evaluation_strategy and save_strategy: This is not a hyperparameter to tune, but it is used to select the evaluation pass at the end of each epoch, and the save_strategy corresponds to saving the model checkpoints after each epoch.
All other parameters are used as stated in question 2.

The output of the both the Models:
These are the outputs generated by taking a random sentence and fine-tuned model of question 2.
For IndicNER fine-tuned Model:

In [10]:

```
from transformers import AutoTokenizer, AutoModelForTokenClassification
import torch
tokenizer = AutoTokenizer.from_pretrained("finetunedINDIC_ner_model")
model = AutoModelForTokenClassification.from_pretrained("finetunedINDIC_ner_model")
```

In [12]:

```
sentence = 'अभिनेत्री सोहा अली खान से उनकी मां अभिनेत्री शर्मिला टैगोर खासी नाराज़ हैं.'

predicted_labels = get_predictions_indicner(sentence=sentence,
                                    tokenizer=tokenizer,
                                    model=model
                                    )
print(predicted_labels)
for index in range(len(sentence.split(' '))):
  print( sentence.split(' ')[index] + '\t' + predicted_labels[index] )
predicted_labels = [i.strip(',') for i in predicted_labels]
# for i in predicted_labels:
#     print(i,end=" ")
```

```
['O', 'B-PER', 'I-PER', 'I-PER', 'O', 'O', 'O', 'O', 'B-PER', 'I-PER', 'O', 'O', 'O', 'O']
अभिनेत्री  O
सोहा    B-PER
अली     I-PER
खान     I-PER
से      O
उनकी    O
मां      O
अभिनेत्री  O
शर्मिला  B-PER
टैगोर   I-PER
खासी    O
नाराज़   O
हैं.     O
```

Figure 1: IndicNER fine-tuned Model

In [11]:

```
from transformers import AutoTokenizer, AutoModelForTokenClassification
import torch
tokenizer = AutoTokenizer.from_pretrained("finetunedBERT_ner_model")
model = AutoModelForTokenClassification.from_pretrained("finetunedBERT_ner_model")
```

In [12]:

```
sentence = 'लोकसभा में कांग्रेस नेता मलिक्कार्जुन खड़गे ने कहा कि प्रधानमंत्री नरेंद्र मोदी भारत के साथ वही करना चाहते हैं'

predicted_labels = get_predictions(sentence=sentence,
                                   tokenizer=tokenizer,
                                   model=model
                                   )
print(predicted_labels)
for index in range(len(sentence.split(' '))):
  print( sentence.split(' ')[index] + '\t' + predicted_labels[index] )
predicted_labels = [i.strip(',') for i in predicted_labels]
```

```
['O', 'O', 'B-LOC', 'O', 'B-PER', 'I-PER', 'O', 'O', 'O', 'O', 'B-PER', 'I-PER', 'B-ORG', 'O', 'O', 'O', 'O', 'O', 'O']
लोकसभा  O
में      O
कांग्रेस    B-LOC
नेता     O
मलिक्कार्जुन       B-PER
खड़गे    I-PER
ने       O
कहा      O
कि       O
प्रधानमंत्री  O
नरेंद्र     B-PER
मोदी     I-PER
भारत     B-ORG
के       O
साथ      O
वही      O
करना     O
चाहते    O
हैं      O
```

Figure 2: IndicBERT fine-tuned Model