Computational Linguistics for Indian Languages

(CS689A)

Assignment 2

SUBMITTED BY:

PRATIBHA GUPTA
231110038

QUESTION 2:

COMPARING MACRO F1-SCORE FOR INDIC BERT AND INDIC NER-

VALIDATION SET:

- MACRO F-SCORE FOR INDIC BERT: 0.6662
- MACRO F1-SCORE FOR 25 SENTENCES USING IndicNER: 0.7761

TEST SET:

- MACRO F-SCORE FOR INDIC BERT: 0.5357142857142857
- MACRO F1-SCORE FOR 25 SENTENCES USING IndicNER: 0.5614035087719298

From this we can observe that **IndicNER gives a better macro F1 score as compared to IndicBERT.** The higher F1-scores of IndicNER suggest that it's better suited for named entity recognition tasks, which aligns with its design and purpose. IndicNER have specialized architectures or training methodologies optimized for NER tasks, enabling it to capture nuances in named entity recognition better than IndicBERT. Since, IndicBERT is a strong language model; it might not be as effective in capturing specific NER-related features compared to a model explicitly designed for NER tasks like IndicNER. This observation suggests that although pre-trained language models like BERT can perform reasonably well across various NLP tasks, task-specific models like IndicNER might still outperform them in their respective domains.

QUESTION 4:

METRICS OUTPUT OF MANUALLY MARKED SENTENCES AND CHATGPT:

```
MACRO F1-SCORE FOR 25 SENTENCES:
 0.47542807823312727
=======================*****==================================
CLASSWISE PRECISION, RECALL AND F1-SCORE FOR 25 SENTENCES:

O  :  [0.8168168168168168, 0.9477351916376306, 0.8774193548387097]
B_PER  :  [1.0, 0.7333333333333333, 0.846153846153846]
I_PER  :  [1.0, 0.8181818181818182, 0.9]
B_LOC  :  [0.7692307692307693, 1.0, 0.8695652173913044]
I_LOC  :  [0.3333333333333333, 1.0, 0.5]
B_ORG  :  [1.0, 0.16666666666666666, 0.2857142857142857]
I_ORG  :  [0, 0.0, 0]
B_MISC  :  [0.0, 0.0, 0]
I_MISC  :  [0.0, 0.0, 0]
```

METRICS OUTPUT OF MANUALLY MARKED SENTENCES AND TAGGING GIVEN BY Indic-BERT:

```
MACRO F1-SCORE FOR 25 SENTENCES USING IndicBERT:
 0.47190975693506293
CLASSWISE PRECISION, RECALL AND F1-SCORE FOR 25 SENTENCES:

O  :  [0.796969696969697, 0.9163763066202091, 0.8525121555915721]
B_PER  :  [0.7368421052631579, 0.9333333333333333, 0.8235294117647058]
I_PER  :  [0.4166666666666667, 0.45454545454545453, 0.43478260869565216]
B_LOC  :  [0.25, 0.3, 0.2727272727272727]
I_LOC  :  [0.5, 1.0, 0.6666666666666666]
B_ORG  :  [0.4, 0.3333333333333333, 0.3636363636363636]
I_ORG  :  [1.0, 0.7142857142857143, 0.8333333333333333]
B_MISC  :  [0, 0.0, 0]
I_MISC  :  [0, 0.0, 0]
```

METRICS OUTPUT OF MANUALLY MARKED SENTENCES AND TAGGING GIVEN BY Indic-NER:

```
MACRO F1-SCORE FOR 25 SENTENCES USING IndicNER:
 0.2833128469036034
CLASSWISE PRECISION, RECALL AND F1-SCORE FOR 25 SENTENCES:

O  :  [0.7648809523809523, 0.8954703832752613, 0.8250401284109148]
B_PER  :  [0.3684210526315789, 0.4666666666666667, 0.4117647058823529]
I_PER  :  [0.16666666666666666, 0.18181818181818182, 0.17391304347826086]
B_LOC  :  [0.2222222222222222, 0.2, 0.21052631578947370]
I_LOC  :  [0, 0.0, 0]
B_ORG  :  [0.375, 0.5, 0.42857142857142855]
I_ORG  :  [0.6, 0.42857142857142855, 0.5]
B_MISC  :  [0, 0.0, 0]
I_MISC  :  [0, 0.0, 0]
```

QUESTION 5:

From the comparisons I can infer that accuracy of model and precision of predicting the tags highly depends on parameters like learning rate and batch size.

- Increasing the batch size decreases the precision of predictions made by the models.

- Also smaller learning rate gives better accuracy as compared with increasing the learning rate.

Hyper parameters setting to improve the performance of models:

The hyper parameters that I have changed:

- **per_device_train_batch_size and per_device_eval_batch_size -** These parameters define the number of training samples and evaluation samples, respectively processed simultaneously on each device (GPU or CPU) during training. Larger batch sizes require more memory, and if the batch size exceeds the available memory, training may fail. Smaller batch sizes generalize better and help prevent overfitting, especially with limited data.

- **Num_train_epochs -** It directly controls how many times the model iterates over the entire training dataset. Increasing this parameter allows the model to see the training data more times, potentially leading to better convergence and improved performance.

- **Learning_rate -** The learning rate determines the size of the step taken in the direction opposite to the gradient during optimization. A higher learning rate means larger steps, potentially leading to faster convergence but with the risk of overshooting the optimal solution.

Optimal values chosen by me:

Optimal results are obtained on following arguments: (for both the models)

- per_device_train_batch_size - 8
- Per_device_eval_batch_size - 8
- Num_train_epochs - 3
- Learning_rate - 5e-5

OUTPUT FOR BOTH THE MODELS -

OUTPUTS FOR INDIC-BERT (ARGUMENTS 1):

```
batch_size=8
args=TrainingArguments(
    output_dir='output_dir',
    per_device_train_batch_size=batch_size,
    per_device_eval_batch_size=batch_size,
    num_train_epochs=3,
    evaluation_strategy = "epoch",
    learning_rate=5e-5
)
```

```
***** eval metrics *****
  epoch                     =        3.0
  eval_LOC_f1               =     0.7224
  eval_LOC_number           =      10213
  eval_LOC_precision        =     0.7169
  eval_LOC_recall           =      0.728
  eval_ORG_f1               =     0.5604
  eval_ORG_number           =       9786
  eval_ORG_precision        =     0.5707
  eval_ORG_recall           =     0.5504
  eval_PER_f1               =     0.7082
  eval_PER_number           =      10568
  eval_PER_precision        =      0.715
  eval_PER_recall           =     0.7016
  eval_loss                 =     0.2606
  eval_overall_accuracy     =     0.9206
  eval_overall_f1           =     0.6662
  eval_overall_precision    =     0.6705
  eval_overall_recall       =      0.662
  eval_runtime              = 0:04:19.63
  eval_samples_per_second   =     51.842
  eval_steps_per_second     =      3.243
```

OUTPUTS FOR INDIC-BERT (ARGUMENTS 2):

```
batch_size=6
args=TrainingArguments(
    output_dir='output_dir',
    per_device_train_batch_size=batch_size,
    per_device_eval_batch_size=batch_size,
    num_train_epochs=3,
    evaluation_strategy = "epoch",
    learning_rate=4e-5
)
```

```
***** eval metrics *****
  epoch                     =        3.0
  eval_LOC_f1               =     0.7227
  eval_LOC_number           =      10213
  eval_LOC_precision        =     0.7183
  eval_LOC_recall           =     0.7272
  eval_ORG_f1               =     0.5614
  eval_ORG_number           =       9786
  eval_ORG_precision        =     0.5665
  eval_ORG_recall           =     0.5564
  eval_PER_f1               =     0.7064
  eval_PER_number           =      10568
  eval_PER_precision        =     0.7162
  eval_PER_recall           =     0.6969
  eval_loss                 =     0.2626
  eval_overall_accuracy     =     0.9202
  eval_overall_f1           =     0.6657
  eval_overall_precision    =     0.6693
  eval_overall_recall       =     0.6621
  eval_runtime              = 0:04:33.20
  eval_samples_per_second   =     49.267
  eval_steps_per_second     =      4.107
```

OUTPUTS FOR INDIC-BERT (ARGUMENTS 3):

```
batch_size=16
args=TrainingArguments(
    output_dir='output_dir',
    per_device_train_batch_size=batch_size,
    per_device_eval_batch_size=batch_size,
    num_train_epochs=3,
    evaluation_strategy = "epoch",
    learning_rate=5e-2
)
```

```
***** eval metrics *****
  epoch                    =        3.0
  eval_LOC_f1              =        0.0
  eval_LOC_number          =      10213
  eval_LOC_precision       =        0.0
  eval_LOC_recall          =        0.0
  eval_ORG_f1              =        0.0
  eval_ORG_number          =       9786
  eval_ORG_precision       =        0.0
  eval_ORG_recall          =        0.0
  eval_PER_f1              =        0.0
  eval_PER_number          =      10568
  eval_PER_precision       =        0.0
  eval_PER_recall          =        0.0
  eval_loss                =     0.7806
  eval_overall_accuracy    =     0.8204
  eval_overall_f1          =        0.0
  eval_overall_precision   =        0.0
  eval_overall_recall      =        0.0
  eval_runtime             = 0:03:59.08
  eval_samples_per_second  =     56.299
  eval_steps_per_second    =      1.761
```

OUTPUTS FOR INDIC-NER (ARGUMENTS 1):

```
args=TrainingArguments(
    output_dir='output_dir',
    per_device_train_batch_size=8,
    per_device_eval_batch_size=8,
    num_train_epochs=3,
    evaluation_strategy = "epoch",
    learning_rate=5e-5)
```

```
***** eval metrics *****
  epoch                    =        3.0
  eval_LOC_f1              =     0.8315
  eval_LOC_number          =      10213
  eval_LOC_precision       =     0.8116
  eval_LOC_recall          =     0.8523
  eval_ORG_f1              =     0.6783
  eval_ORG_number          =       9786
  eval_ORG_precision       =     0.6715
  eval_ORG_recall          =     0.6853
  eval_PER_f1              =     0.8122
  eval_PER_number          =      10568
  eval_PER_precision       =     0.8006
  eval_PER_recall          =     0.8242
  eval_loss                =     0.2078
  eval_overall_accuracy    =     0.9459
  eval_overall_f1          =     0.7761
  eval_overall_precision   =     0.7635
  eval_overall_recall      =     0.7891
  eval_runtime             = 0:05:11.69
  eval_samples_per_second  =     43.183
  eval_steps_per_second    =      2.701
```

OUTPUTS FOR INDIC-NER (ARGUMENTS 2):

```
args=TrainingArguments(
    output_dir='output_dir',
    per_device_train_batch_size=8,
    per_device_eval_batch_size=8,
    num_train_epochs=3,
    evaluation_strategy = "epoch",
    learning_rate=5e-3)
```

```
***** eval metrics *****
  epoch                   =        3.0
  eval_LOC_f1             =     0.8315
  eval_LOC_number         =      10213
  eval_LOC_precision      =     0.8116
  eval_LOC_recall         =     0.8523
  eval_ORG_f1             =     0.6783
  eval_ORG_number         =       9786
  eval_ORG_precision      =     0.6715
  eval_ORG_recall         =     0.6853
  eval_PER_f1             =     0.8122
  eval_PER_number         =      10568
  eval_PER_precision      =     0.8006
  eval_PER_recall         =     0.8242
  eval_loss               =     0.2078
  eval_overall_accuracy   =     0.9459
  eval_overall_f1         =     0.7761
  eval_overall_precision  =     0.7635
  eval_overall_recall     =     0.7891
  eval_runtime            = 0:05:11.69
  eval_samples_per_second =     43.183
  eval_steps_per_second   =      2.701
```

OUTPUTS FOR INDIC-NER (ARGUMENTS 3):

```
batch_size=8
args=TrainingArguments(
    output_dir='output_dir',
    per_device_train_batch_size=batch_size,
    per_device_eval_batch_size=batch_size,
    num_train_epochs=3,
    evaluation_strategy = "epoch",
    weight_decay=0.1,
    learning_rate=4e-5)
```

```
***** eval metrics *****
  epoch                   =        3.0
  eval_LOC_f1             =     0.8328
  eval_LOC_number         =      10213
  eval_LOC_precision      =     0.8136
  eval_LOC_recall         =      0.853
  eval_ORG_f1             =      0.683
  eval_ORG_number         =       9786
  eval_ORG_precision      =     0.6776
  eval_ORG_recall         =     0.6884
  eval_PER_f1             =      0.814
  eval_PER_number         =      10568
  eval_PER_precision      =     0.8022
  eval_PER_recall         =     0.8262
  eval_loss               =     0.2032
  eval_overall_accuracy   =     0.9464
  eval_overall_f1         =     0.7787
  eval_overall_precision  =     0.7668
  eval_overall_recall     =      0.791
  eval_runtime            = 0:05:07.58
  eval_samples_per_second =      43.76
  eval_steps_per_second   =      2.737
```

OUTPUTS FOR INDIC-NER (ARGUMENTS 4):

```
batch_size=6
args=TrainingArguments(
    output_dir='output_dir',
    per_device_train_batch_size=batch_size,
    per_device_eval_batch_size=batch_size,
    num_train_epochs=3,
    evaluation_strategy = "epoch",
    learning_rate=5e-7)
```

```
***** eval metrics *****
  epoch                   =        3.0
  eval_LOC_f1             =     0.5742
  eval_LOC_number         =      10213
  eval_LOC_precision      =     0.5671
  eval_LOC_recall         =     0.5815
  eval_ORG_f1             =     0.3439
  eval_ORG_number         =       9786
  eval_ORG_precision      =     0.3465
  eval_ORG_recall         =     0.3413
  eval_PER_f1             =     0.5299
  eval_PER_number         =      10568
  eval_PER_precision      =     0.4378
  eval_PER_recall         =     0.6712
  eval_loss               =     0.3268
  eval_overall_accuracy   =     0.9002
  eval_overall_f1         =     0.4896
  eval_overall_precision  =     0.4509
  eval_overall_recall     =     0.5356
  eval_runtime            = 0:05:56.19
  eval_samples_per_second =     37.788
  eval_steps_per_second   =       3.15
```

CONCLUSION:

From the above outputs we can conclude that increasing the batch size significantly decreases the accuracy as smaller batch size gives better generalization and also helps reduce overfitting. And lower learning rate gives better convergence and helps models gain better accuracy. So, we can alter the hyper parameter for the training arguments and tune it to the most optimal one.