Computational Linguistics for Indian Languages

(CS689A)

ASSIGNMENT 2

SUBMITTED BY: PRASHIK GANER

ROLL NO: 231110037

QUESTION 2

Macro F1 scores for validation set:

IndicBERT: Macro-F1 Score: 0.3558

IndicNER: Macro-F1 Score: 0.6092

Macro F1 scores for test set:

IndicBERT: Macro-F1 Score: 0.5263157894736842

IndicNER: Macro-F1 Score: 0.5161290322580645

Observations:

1. Performance Comparison: IndicBERT outperformed IndicNER in terms of macro-F1 score, suggesting its superiority in certain NLP tasks.
2. Task Suitability: The choice between IndicBERT and IndicNER should consider the specific requirements of the task. Despite its lower macro-F1 score, IndicNER may still be more suitable for named entity recognition tasks.
3. Further Analysis Opportunities: It's crucial to conduct further analysis to understand the factors contributing to the performance differences between IndicBERT and IndicNER. This analysis could include examining specific instances where each model succeeded or struggled and identifying potential areas for improvement.

In summary, while IndicBERT achieved a higher macro-F1 score compared to IndicNER, task-specific considerations and opportunities for improvement should guide the selection and refinement of NLP models.

QUESTION 3

Refer ‘CL - Assignment 2, question 3.pdf’

QUESTION 4

Refer ‘CL_A2_Q4_FINAL.ipynb’ file

Metrics output of:

Metrices	Output
Manually marked sentences and CHATGPT	metrices for class B-PER Precision: 1.0 Recall: 0.15384615384615385 metrices for class I-PER Precision: 1.0 Recall: 0.3333333333333333 metrices for class B-MISC Precision: 0.0 Recall: 0 metrices for class I-MISC Precision: 0.0 Recall: 0 metrices for class O Precision: 0.9255663430420712 Recall: 0.9255663430420712 metrices for class B-ORG Precision: 0 Recall: 0.0 metrices for class I-ORG Precision: 0 Recall: 0.0 metrices for class B-LOC Precision: 0.7142857142857143 Recall: 0.5 metrices for class I-LOC Precision: 0 Recall: 0

Manually marked sentences and Tagging given by IndicBERT	metrices for class B-PER Precision: 1.0 Recall: 0.15384615384615385 metrices for class I-PER Precision: 1.0 Recall: 0.3333333333333333 metrices for class B-MISC Precision: 0.0 Recall: 0 metrices for class I-MISC Precision: 0.0 Recall: 0 metrices for class O Precision: 0.9255663430420712 Recall: 0.9255663430420712 metrices for class B-ORG Precision: 0 Recall: 0.0 metrices for class I-ORG Precision: 0 Recall: 0.0 metrices for class B-LOC Precision: 0.7142857142857143 Recall: 0.5 metrices for class I-LOC Precision: 0 Recall: 0

Manually marked sentences and Tagging given by IndicNER	metrices for class B-PER Precision: 1.0 Recall: 0.3333333333333333 metrices for class I-PER Precision: 1.0 Recall: 1.0 metrices for class B-MISC Precision: 0.0 Recall: 0 metrices for class I-MISC Precision: 0.0 Recall: 0 metrices for class O Precision: 0.970873786407767 Recall: 0.9345794392523364 metrices for class B-ORG Precision: 0 Recall: 0.0 metrices for class I-ORG Precision: 0 Recall: 0 metrices for class B-LOC Precision: 0.8571428571428571 Recall: 0.6666666666666666

QUESTION 5

In this assignment, we conducted a comparison between two approaches for Named Entity Recognition (NER) in a specific language using the Naamapadam corpus. The comparison involved fine-tuning pre-trained language models (IndicBERT and IndicNER) and utilizing ChatGPT for NER.

Brief Introduction to IndicBERT and IndicNER:

IndicBERT: IndicBERT is a variant of the BERT (Bidirectional Encoder Representations from Transformers) model specifically trained on Indic languages. BERT is a powerful pre-trained language model developed by Google, known for its ability to understand context and semantics in natural language. IndicBERT extends this capability to Indic languages, making it suitable for various NLP tasks, including Named Entity Recognition (NER).

IndicNER: IndicNER is a Named Entity Recognition model fine-tuned specifically for Indic languages. It leverages deep learning techniques to identify named entities such as persons, organizations, and locations within text. IndicNER is trained on large annotated datasets to accurately recognize named entities in Indic language text.

Usage of IndicBERT and IndicNER:

IndicBERT: To utilize IndicBERT, I have accessed the pre-trained model from the Hugging Face model hub and fine-tuned it according to specific requirements. This fine-tuning process involves providing labeled data and adjusting hyperparameters to optimize performance. Once fine-tuned, IndicBERT can be applied to a variety of NLP tasks, including text classification and question answering.

IndicNER: IndicNER can be employed by loading the pre-trained model and fine-tuning it on annotated NER datasets. During fine-tuning, the model learns to recognize named entities within text based on examples provided in the training data. This enables IndicNER to accurately identify entities in new text inputs.

Hyperparameters in Fine-Tuning IndicBERT and IndicNER:

Hyperparameters are parameters whose values are set before the learning process begins. Tuning these parameters can significantly impact the training and fine-tuning of these models.

Learning Rate:

Learning rate controls the step size during the optimization process. It determines how much the model's parameters are adjusted during each update.
Tuning: Learning rate can be tuned to find the optimal balance between convergence speed and stability. Too high a learning rate may cause instability or divergence, while too low a learning rate may result in slow convergence.

Batch Size:

Batch size refers to the number of samples processed before updating the model's parameters. Larger batch sizes can accelerate training but may require more memory.
Tuning: Batch size can be adjusted based on available computational resources. It's essential to find a balance between computational efficiency and model performance.

Number of Epochs:

The number of epochs determines how many times the entire dataset is passed through the model during training. One epoch is a single forward and backward pass of all training samples.
Tuning: The number of epochs should be chosen to prevent overfitting while allowing the model to learn the underlying patterns in the data. Early stopping can be employed to avoid training for too many epochs.

I’ve fine tuned the model 4 times while changing the three hyperparameters described above, with hyperparameters being:

Precision , Recall , F1 Score of fine tune on Indic-Bert:

Hyperparameter	Value
Batch Size	16
Number of Epochs	3
Learning Rate	5e-7

```
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-7)
```

[1875/1875 58:54, Epoch 3/3]

Epoch	Training Loss	Validation Loss	Loc Precision	Loc Recall	Loc F1	Loc Number	Org Precision	Org Recall	Org F1	Org Number	Per Precision	Per Recall	Per F1	Per Number	Overall Precision	Overall Recall	Overall F1	Overall Accuracy
1	1.240200	0.764581	0.000000	0.000000	0.000000	10213	0.000000	0.000000	0.000000	9786	0.000000	0.000000	0.000000	10568	0.000000	0.000000	0.000000	0.820403
2	0.749300	0.702868	0.000000	0.000000	0.000000	10213	0.000000	0.000000	0.000000	9786	0.000000	0.000000	0.000000	10568	0.000000	0.000000	0.000000	0.820403
3	0.699500	0.686462	0.000000	0.000000	0.000000	10213	0.000000	0.000000	0.000000	9786	0.000000	0.000000	0.000000	10568	0.000000	0.000000	0.000000	0.820403

Hyperparameter	Value
Batch Size	8
Number of Epochs	3
Learning Rate	5e-5

```
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)
```

[3750/3750 57:40, Epoch 3/3]

Epoch	Training Loss	Validation Loss	Loc Precision	Loc Recall	Loc F1	Loc Number	Org Precision	Org Recall	Org F1	Org Number	Per Precision	Per Recall	Per F1	Per Number	Overall Precision	Overall Recall	Overall F1	Overall Accuracy
1	0.319000	0.298102	0.690034	0.665035	0.677304	10213	0.594140	0.418557	0.491127	9786	0.699504	0.614118	0.654036	10568	0.667999	0.568522	0.614259	0.910376
2	0.239200	0.261304	0.720679	0.711152	0.715884	10213	0.582748	0.526058	0.552954	9786	0.734439	0.678842	0.705547	10568	0.682880	0.640724	0.661131	0.919216
3	0.202200	0.259096	0.713632	0.732498	0.722942	10213	0.577398	0.559166	0.568136	9786	0.723214	0.705148	0.714067	10568	0.674233	0.667550	0.670875	0.920830

Hyperparameter	Value
Batch Size	8
Number of Epochs	3
Learning Rate	2e-6

```
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=2e-6)
```

[3750/3750 57:05, Epoch 3/3]

Epoch	Training Loss	Validation Loss	Loc Precision	Loc Recall	Loc F1	Loc Number	Org Precision	Org Recall	Org F1	Org Number	Per Precision	Per Recall	Per F1	Per Number	Overall Precision	Overall Recall	Overall F1	Overall Accuracy
1	0.601800	0.559947	0.596390	0.080877	0.142438	10213	0.733333	0.001124	0.002245	9786	0.171524	0.038986	0.063531	10568	0.328511	0.040861	0.072682	0.828810
2	0.486100	0.484523	0.502328	0.348673	0.411629	10213	0.476802	0.068261	0.119424	9786	0.490062	0.352290	0.409909	10568	0.494312	0.260150	0.340893	0.857540
3	0.464400	0.466076	0.491430	0.401449	0.441906	10213	0.377145	0.112303	0.173071	9786	0.520090	0.416446	0.462533	10568	0.486840	0.314064	0.381816	0.863696

Hyperparameter	Value
Batch Size	6
Number of Epochs	3
Learning Rate	5e-7

```
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
)
```

[5001/5001 1:00:10, Epoch 3/3]

Epoch	Training Loss	Validation Loss	Loc Precision	Loc Recall	Loc F1	Loc Number	Org Precision	Org Recall	Org F1	Org Number	Per Precision	Per Recall	Per F1	Per Number	Overall Precision	Overall Recall	Overall F1	Overall Accuracy
1	0.667300	0.649048	0.000000	0.000000	0.000000	10213	0.000000	0.000000	0.000000	9786	0.000000	0.000000	0.000000	10568	0.000000	0.000000	0.000000	0.820403
2	0.615300	0.610860	0.000000	0.000000	0.000000	10213	0.000000	0.000000	0.000000	9786	0.000000	0.000000	0.000000	10568	0.000000	0.000000	0.000000	0.820403
3	0.594100	0.600951	0.000000	0.000000	0.000000	10213	0.000000	0.000000	0.000000	9786	0.000000	0.000000	0.000000	10568	0.000000	0.000000	0.000000	0.820410

Observations w.r.t data:

1. The configuration with a batch size of 16 and a learning rate of 5e-7 achieved an overall accuracy of 0.820403.
2. The configuration with a batch size of 8 and a learning rate of 5e-5 achieved the highest overall accuracy of 0.920830.
3. Another configuration with a batch size of 8 and a learning rate of 2e-6 achieved an overall accuracy of 0.863696.
4. The configuration with a batch size of 6 and a learning rate of 5e-7 achieved an overall accuracy of 0.820410.

Conclusion:

1. The configuration with a batch size of 8 and a learning rate of 5e-5 achieved the highest overall accuracy among all configurations, reaching 92.08%.
2. Configurations with smaller batch sizes (6 and 8) generally achieved similar accuracies compared to the configuration with a batch size of 16.
3. The configuration with a learning rate of 5e-5 consistently outperformed configurations with lower learning rates (5e-7 and 2e-6).

We can see that, for fine-tuning IndicBERT, a batch size of 8 with a learning rate of 5e-5 resulted in the highest overall accuracy, emphasizing the importance of tuning hyperparameters for optimal performance

Output for IndicBERT model:

```python
# let us try with some example sentences here
# sentence = 'लगातार हमलावर हो रहे शिवपाल और राजभर को सपा की दो टूक, चिट्ठी जारी कर कहा- जहां जाना चाहें जा सकते हैं'
sentences = ['इसके अलावा मानेसर-बावल इनवेस्टमेंट रीजन के लिए मास्टर प्लान  तैयार किया', ' महोदय, हमारे देश में सारा कामकाज बुल्गारियाई में ही किया जाता है हम वकील सारे दस्तावेज केवल बुल्गारियाई में ही तैयार करते हैं और न्यायालय में भी सभी कार्यवाह

pred_labels = []
for sentence in sentences:
    predicted_labels = get_predictions(sentence=sentence,
                                       tokenizer=tokenizer,
                                       model=model
                                       )
    pred_labels.append(predicted_labels)
```

Python

```python
print(pred_labels)
```

Python

```
[['LABEL_0', 'LABEL_0', 'LABEL_5', 'LABEL_0', 'LABEL_0', 'LABEL_0', 'LABEL_0', 'LABEL_0', 'LABEL_0', 'LABEL_0', 'LABEL_0'], ['LABEL_0', 'LABEL_0', 'LABEL_0', 'LABEL_0', 'LABEL_0', 'LABEL_0', 'LABEL_0', 'LABEL_0'
```

.

Precision , Recall , F1 Score of fine tune on Indic-NER:

Hyperparameter	Value
Batch Size	16
Number of Epochs	3
Learning Rate	5e-7

```python
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-7)
```

[1875/1875 1:06:04, Epoch 3/3]

Epoch	Training Loss	Validation Loss	Loc Precision	Loc Recall	Loc F1	Loc Number	Org Precision	Org Recall	Org F1	Org Number	Per Precision	Per Recall	Per F1	Per Number	Overall Precision	Overall Recall	Overall F1	Overall Accuracy
1	6.928800	2.577833	0.000169	0.000783	0.000278	10213	0.016926	0.011241	0.013509	9786	0.124781	0.208460	0.156114	10568	0.032456	0.075932	0.045474	0.076111
2	1.635000	0.648141	0.000209	0.000098	0.000133	10213	0.017225	0.005722	0.008591	9786	0.080736	0.068887	0.074343	10568	0.046025	0.025681	0.032967	0.827768
3	0.655200	0.585783	0.000164	0.000098	0.000123	10213	0.024151	0.009810	0.013952	9786	0.101530	0.118660	0.109429	10568	0.060286	0.044198	0.051003	0.830676

Hyperparameter	Value
Batch Size	8
Number of Epochs	3
Learning Rate	5e-5

```python
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)
```

[3750/3750 1:11:56, Epoch 3/3]

Epoch	Training Loss	Validation Loss	Loc Precision	Loc Recall	Loc F1	Loc Number	Org Precision	Org Recall	Org F1	Org Number	Per Precision	Per Recall	Per F1	Per Number	Overall Precision	Overall Recall	Overall F1	Overall Accuracy
1	0.153100	0.170914	0.810615	0.855380	0.832396	10213	0.683332	0.677396	0.680351	9786	0.806352	0.838475	0.822100	10568	0.769886	0.792554	0.781056	0.947092
2	0.103200	0.182538	0.817916	0.850191	0.833741	10213	0.673960	0.698753	0.686133	9786	0.804627	0.829296	0.816775	10568	0.767202	0.794484	0.780605	0.946835
3	0.074200	0.207841	0.811579	0.852345	0.831463	10213	0.671473	0.685265	0.678299	9786	0.800625	0.824186	0.812235	10568	0.763516	0.789119	0.776106	0.945863

Hyperparameter	Value
Batch Size	8
Number of Epochs	3
Learning Rate	2e-6

```python
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=2e-6)
```

[3750/3750 1:10:57, Epoch 3/3]

Epoch	Training Loss	Validation Loss	Loc Precision	Loc Recall	Loc F1	Loc Number	Org Precision	Org Recall	Org F1	Org Number	Per Precision	Per Recall	Per F1	Per Number	Overall Precision	Overall Recall	Overall F1	Overall Accuracy
1	0.286600	0.221403	0.703680	0.765691	0.733377	10213	0.592859	0.648171	0.619282	9786	0.736248	0.820685	0.776177	10568	0.679805	0.747080	0.711856	0.937712
2	0.178700	0.196081	0.792105	0.819250	0.805449	10213	0.642465	0.681790	0.661544	9786	0.777179	0.832702	0.803983	10568	0.738713	0.779893	0.758745	0.944022
3	0.173900	0.192840	0.792441	0.831391	0.811449	10213	0.649062	0.685673	0.666865	9786	0.784305	0.835068	0.808891	10568	0.743724	0.786011	0.764283	0.944817

Hyperparameter	Value
Batch Size	6
Number of Epochs	3
Learning Rate	5e-7

```python
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)
```

[5001/5001 1:18:25, Epoch 3/3]

Epoch	Training Loss	Validation Loss	Loc Precision	Loc Recall	Loc F1	Loc Number	Org Precision	Org Recall	Org F1	Org Number	Per Precision	Per Recall	Per F1	Per Number	Overall Precision	Overall Recall	Overall F1	Overall Accuracy
1	0.621400	0.508421	0.000569	0.000392	0.000464	10213	0.028395	0.014102	0.018845	9786	0.127784	0.196537	0.154873	10568	0.078831	0.072595	0.075584	0.833736
2	0.378400	0.354153	0.480334	0.511799	0.495568	10213	0.248652	0.221439	0.234258	9786	0.333690	0.560749	0.418399	10568	0.356569	0.435764	0.392209	0.883897
3	0.320900	0.326751	0.567131	0.581514	0.574233	10213	0.346473	0.341304	0.343869	9786	0.437812	0.671177	0.529941	10568	0.450858	0.535610	0.489593	0.900224

Observations w.r.t data:

1. The configuration with a batch size of 16 and a learning rate of 5e-7 achieved an overall accuracy of 0.830676.
2. The configuration with a batch size of 8 and a learning rate of 5e-5 achieved the highest overall accuracy of 0.9458.
3. Another configuration with a batch size of 8 and a learning rate of 2e-6 also achieved high accuracy, with an overall accuracy of 0.9448.
4. The configuration with a batch size of 6 and a learning rate of 5e-7 achieved an overall accuracy of 0.900.

Conclusion:

Here, the choice of hyperparameters such as batch size, number of epochs, and learning rate significantly affects the performance of the Indic NER model, with higher accuracy generally observed with smaller batch sizes and moderate learning rates.

Output for IndicNER model:

```python
# let us try with some example sentences here
# sentence = 'लगातार हमलावर हो रहे शिवपाल और राजभर को सपा की दो टूक, चिट्ठी जारी कर कहा- जहां जाना चाहें जा सकते हैं'
sentences = ['इसके अलावा मानेसर-बावल इनवेस्टमेंट रीजन के लिए मास्टर प्लान  तैयार किया', ' महोदय, हमारे देश में सारा कामकाज बुल्गारियाई में ही किया जाता है हम वकील सारे दस्तावेज केवल बुल्गारियाई में ही तैयार करते हैं और न्यायालय में भी सभी कार्यवाह

pred_labels = []
for sentence in sentences:
    predicted_labels = get_predictions(sentence=sentence,
                                       tokenizer=tokenizer,
                                       model=model
                                       )
    pred_labels.append(predicted_labels)
```

33] Python

```python
print(pred_labels)
```

34] Python

```
[['B-LOC', 'B-LOC', 'I-PER', 'B-LOC', 'I-PER', 'O', 'O', 'B-LOC', 'B-LOC', 'B-LOC', 'B-LOC', 'B-LOC', 'B-LOC'], ['B-LOC', 'B-LOC', 'B-LOC', 'B-LOC', 'B-LOC', 'B-LOC', 'B-LOC', 'B-LOC', 'B-LOC', 'B-LOC', 'B-LOC',
```

Optimal results are obtained on following arguments:

The best outcomes are achieved with the following settings(for both models):

- [ ] per_device_train_batch_size: 8
- [ ] per_device_eval_batch_size: 8
- [ ] Num_train_epochs: 3
- [ ] Learning_rate: 5e-5

Overall Comparison of IndicBERT and IndicNER:

IndicBERT:

- Best Configuration: Batch size 8, Learning rate 5e-5, Overall accuracy 0.920830.
- Observations: IndicBERT achieved the highest overall accuracy among the configurations, indicating its effectiveness in capturing language features and nuances for the given task.
- Hyperparameters: Batch size ranged from 6 to 16, learning rates varied from 5e-7 to 5e-5, and the number of epochs was constant at 3.

IndicNER:

- Best Configuration: Batch size 8, Learning rate 5e-5, Overall accuracy 0.9458.
- Observations: IndicNER achieved the highest overall accuracy among the configurations, surpassing IndicBERT. This suggests that IndicNER is more suitable for named entity recognition tasks in Indic languages.
- Hyperparameters: Similar to IndicBERT, the batch size ranged from 6 to 16, learning rates varied from 5e-7 to 5e-5, and the number of epochs was constant at 3.

Observations and Insights:

- Both IndicBERT and IndicNER performed well across different configurations, with IndicNER slightly outperforming IndicBERT.
- Hyperparameters such as batch size and learning rate had a notable impact on model performance. For both models, a batch size of 8 and a learning rate of 5e-5 resulted in the highest accuracies.
- IndicNER, being specifically fine-tuned for named entity recognition tasks, demonstrated superior performance compared to IndicBERT in this particular task.

Hyperparameters Set for Fine-tuning:

- Batch Size: Ranged from 6 to 16 for both IndicBERT and IndicNER.
- Learning Rate: Varied from 5e-7 to 5e-5 for both models.
- Number of Epochs: Consistently set to 3 for both models.

In summary, while both IndicBERT and IndicNER achieved high accuracies, IndicNER proved to be more effective for named entity recognition tasks in Indic languages. Tuning hyperparameters such as batch size and learning rate is crucial for optimizing the performance of both models.