Question 5: Learning from Comparison

Comparison of Both the Models:

Below are the observations I have made while going through the predictions of NER tags by both the fine-tuned models.

Labelling Problem:

Both the models are not generating B-MISC and I-MISC tags as advised to us for manual tagging. This will reduce the performance parameters such as *Precision, Recall* and *F-Score*.

Model's Comparison:

We are using indicNER and indicBERT models for the NER tagging tasks. Since indicNER is a fine-tuned model using billions of annotated training corpus, this model would be having superior performance over indicBERT model for the same task.

This advantage of indicNER over indicBERT is clearly visible in our analysis which will be shown in further sections below.

Pre-Processing for Ground Truth:

For creating ground truth tags for the 25 sentences assigned to me, I have manually annotated these sentences as per the given instruction of using BIO format classifying tokens in four categories as ORG(B,I), LOC(B,I), MISC(B,I) and O.

Since annotating NER tags require a decent understanding of the language, I might have misclassified at some places which also might impact performance of the models.

Comparison of Manual and ChatGPT tagging:

Though ChatGPT is not specifically trained for this task, while giving proper prompt the results given by ChatGPT have identified *ORG* and *O* tags correctly. But it failed for correctly classifying *LOC* tags and at many places predicted them as *PER* tags despite giving multiple prompts to rectify the issue.

Evaluation parameters of ChatGPT against manual NER tags is as below.

Macro-Evaluation Parameters are as below:

```
Precision:      14.39
Recall:         14.96
F1-score:       14.66
```

One might think that maximum NER tags identified by ChatGPT and Manually are matching, then why we are getting such low evaluation parameters. Reasoning behind such low numbers is explained in below snapshot.

Category-wise Precision:

```
Class O:          Precision = 90.16
Class B-ORG:      Precision = 18.18
Class I-ORG:      Precision = 30.00
Class B-MISC:     Precision = 0.00
Class B-PER:      Precision = 0.00
Class I-PER:      Precision = 0.00
Class B-LOC:      Precision = 5.56
Class I-LOC:      Precision = 0.00
Class I-MISC:     Precision = 0.00
Class B-EVENT:    Precision = 0.00
```

If we see carefully the obtained Precision parameter category-wise, it is evident that because of the some tags not identified by both the methods, aggregation is giving overall small values of Evaluation Parameters.

Fine Tuning of indicNER and indicBERT models:

DataSet Used for Fine-Tuning Task:

For fine tuning both the specified models, I have used *Naampadam* dataset in *Hindi* language. This dataset is already annotated in *token : ner_tag* format.

For training both the models, I have taken 20,000 instances of training sample out of given 985787 instances.

Fine-Tuned indicNER Model:

While fine-tuning indicNER model, I have used two iterations of hyper-parameter tuning and observed the evaluation parameters.

First Set of Hyper-Parameters and Corresponding Result:

```
batch_size = 16
args = TrainingArguments(
    output_di r= '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-5)
```

Epoch	Training Loss	Validation Loss	Overall Precision	Overall Recall	Overall F1	Overall Accuracy
1	0.4518	0.176986	0.767812	0.795368	0.781347	0.947608
2	0.1374	0.176045	0.76874	0.79985	0.783986	0.948285
3	0.1161	0.183262	0.766593	0.798017	0.78199	0.947512

Second Set of Hyper-Parameters and Corresponding Result:

```
batch_size = 8
args = TrainingArguments(
    output_di r= '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-5)
```

Epoch	Training Loss	Validation Loss	Overall Precision	Overall Recall	Overall F1	Overall Accuracy
1	0.1533	0.173247	0.774905	0.794223	0.784445	0.948018
2	0.1157	0.176183	0.771442	0.795368	0.783222	0.947861
3	0.0976	0.188545	0.768434	0.796414	0.782174	0.947236

While analysing the above results, we observed a decline in accuracy as the no of epochs move forward. Explanation behind this behaviour might be from the fact, indicNER is already fine-tuned model on similar to *Naampadam* dataset, further tuning is leading to *over-parameterization,* which is causing *overfitting* of model and in-turn reducing the performance on validation-dataset.

When batch size is reduced, in first epoch it gives better performance as compared to first set of hyper-parameters. But on further epochs, its performance started reducing.

Trade-off:

Based on above observations, as a trade-off, when reducing the batch size of the training data, we should increase the learning rate, so as to compensate the induced noise arising from the smaller batch size.

Analysing above, I will go for below hyper-parameters for indicNER model training.

Batch_size = 12, learning_rate = 1e-6, epochs = 3 (based on available computing and storing infrastructure)

Performance on Test-Dataset:

Below evaluation parameters observed while checking on *test_dataset.*

For first set of hyper-parameters:

Precision:	0.773
Recall:	0.840
F1 - Score:	0.805

For second set of hyper-parameters:

Precision:	0.774
Recall:	0.841
F1 - Score:	0.806

Above two scores of evaluation parameters justify my thought of selecting hyper-parameters.

Fine-Tuned indicBERT Model:

While fine-tuning indicBERT model, I have used two iterations of hyper-parameter tuning and observed the evaluation parameters.

First Set of Hyper-Parameters and Corresponding Result:

```
batch_size = 16
args = TrainingArguments(
    output_di r= '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-5)
```

Epoch	Training Loss	Validation Loss	Overall Precision	Overall Recall	Overall F1	Overall Accuracy
1	0.491	0.357419	0.534969	0.551543	0.543129	0.891837
2	0.3285	0.312674	0.610601	0.579219	0.594497	0.904838
3	0.2888	0.305183	0.620537	0.597049	0.608567	0.907402

Second Set of Hyper-Parameters and Corresponding Result:

```
batch_size = 8
args = TrainingArguments(
    output_di r= '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-5)
```

Epoch	Training Loss	Validation Loss	Overall Precision	Overall Recall	Overall F1	Overall Accuracy
1	0.5049	0.495392	0.45114	0.227762	0.302702	0.858342
2	0.4265	0.434264	0.478402	0.37246	0.418836	0.874081
3	0.4018	0.420833	0.491572	0.399745	0.440928	0.877084

While analysing the above results, we observed a increase in accuracy as the number of epochs move forward when batch_size was set to 16. While it decreased when reducing the batch_size to 8 in next setting.

My reasoning behind observing this behaviour is that the indicBERT model is multipurpose pre-trained model on *Indian Languages*. Since NER tagging is a specific task, to enable the model to do it accurately, we need to fine-tune this model with sufficiently large training dataset. But due to the lack of infrastructure capable to handle that level of computation and storage, we trained this

model on a very small corpus of 20000 instances. This might be the reason behind overall low accuracy.

Talking about the decline in performance in 2nd setting of batch size = 8, reason can be the low learning rate.

Trade-off:

Based on above observations, as a trade-off, when reducing the batch size of the training data, we should increase the learning rate, so as to compensate the induced noise arising from the smaller batch size.

Analysing above, I will go for below hyper-parameters for indicNER model training.

Batch_size = 12, learning_rate = 1e-6, epochs = 3 (based on available computing and storing infrastructure)

Performance on Test-Dataset:

Below evaluation parameters observed while checking on *test_dataset*.

For first set of hyper-parameters:

Precision:	0.655
Recall:	0.636
F1 - Score:	0.645

For second set of hyper-parameters:

Precision:	0.652
Recall:	0.634
F1 - Score:	0.643

Since there is no significant drop in accuracy in both the hyper-parameter settings, *Evaluation parameters* for test_dataset are almost similar.

Comparison of both the Model's Performance on given 25 Sentences in Q1:

Performance of Fine-Tuned indicNER:

As expected, evaluation parameters of indicNER model are the best out of all three methods.

```
Precision:      28.47
Recall:         31.22
F1-score:       29.02
```

Performance of Fine-Tuned indicBERT:

indicBERT performed better than ChatGPT but inferior to indicNER.

Evaluation parameters are as below.

```
Precision:      16.91
Recall:         14.16
F1-score:       14.66
```