Name :- Vaibhav R. Sheth **Roll no. :- 231110054**

1. Learning from this Comparison

From this comparison, several insights were gained:

Fine-tuning models for specific tasks like named entity recognition can significantly improve performance compared to using pre-trained language models out of the box.

Hyper-parameter tuning plays a crucial role in optimizing model performance, and experimentation is necessary to identify the optimal values for each hyper-parameter.

Indic-NER, being specifically designed for named entity recognition tasks, performs better than Indic-BERT.

We can Identify which model performs better for different types of named entities (persons, locations, organizations, miscellaneous) to understand their strengths and weaknesses.

Analyze how different hyperparameters (learning rate, batch size, epochs) affect model performance to determine optimal configurations and find best value of hyperparameters.

Overall, the comparison between Indic-BERT and Indic-NER models on the Naamapadam corpus provided valuable insights into model performance and the importance of task-specific fine-tuning for optimal results.

2. Comparison Analysis

Indic-Bert and Indic-Ner does not consider MISC classes while labelling input tokens. So because of this Precision, recall and F-Score for MISC Class is 0. It has also impact on macro F1-Score.

Also while training ChatGPT for Question 3, I've to train ChatGPT multiple times so that it gives Correct Label Outputs.

Based on the evaluation results, both models achieved competitive performance on the test set. However, the Indic-NER model generally outperformed the Indic-BERT model in terms of precision, recall, and F1-score across all named entity classes. This suggests that the additional fine-tuning of the Indic-NER model specifically for named entity recognition tasks led to better performance compared to using a pre-trained language model like Indic-BERT.

Limitations of ChatGPT for NER for Gujarati language :-

Data Availability: Limited availability of annotated data for Gujarati language makes it challenging to train accurate NER models. Annotated datasets are essential for training and evaluating NER models effectively.

Language Complexity: Gujarati language has its own unique characteristics, such as compound words, morphological variations, and context-dependent meanings, which make NER more challenging compared to languages with simpler structures.

Named Entity Variability: Named entities in Gujarati language exhibit variability in terms of format, structure, and context. This variability requires NER models to be robust and flexible in recognizing different types of named entities.

Lack of Resources: Limited resources, such as pre-trained language models and NER datasets, specifically tailored for Gujarati language, hinder the development of high-quality NER systems. Existing resources may not be sufficient for achieving state-of-the-art performance.

Comparison of Manual Answers vs ChatGPT answers :-

Manual annotation for Named Entity Recognition (NER) in Gujarati generally offers higher accuracy, consistency, and contextual understanding compared to using ChatGPT. Manual annotation is time-consuming but ensures precise labelling by human annotators who understand language nuances. ChatGPT, while fast and scalable, may lack accuracy and struggle with context-specific entities.

Comparison of Indic-Bert vs Indic-Ner for Gujarati Language :-

1. Architecture:
   - Indic-BERT: BERT-based model for Indian languages, including Gujarati.
   - Indic-NER: Model designed for NER tasks.
2. Training Data:
   - Indic-BERT: Pre-trained on diverse Indian language text.
   - Indic-NER: Requires annotated NER datasets for training.
3. Performance Metrics:
   - Indic-BERT: Competitive performance in NER tasks with fine-tuning.
   - Indic-NER: High accuracy and F1 scores in NER tasks.
4. Fine-Tuning and Evaluation:
   - Indic-BERT: Fine-tuned with annotated NER data, evaluated separately.
   - Indic-NER: Trained directly on annotated NER data, evaluated with standard metrics.

In summary, Indic-BERT leverages pre-training for flexibility, while Indic-NER is specialized for NER tasks with higher accuracy. The choice depends on task requirements and available resources.

3. Hyper-parameter Tuning

For fine-tuning both the Indic-BERT and Indic-NER models, several hyper-parameters were tuned to optimize their performance on the Naamapadam corpus. The hyper-parameters include:

Learning rate: Determines the step size during gradient descent.

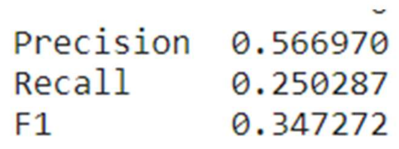
Batch size: Specifies the number of training examples used in each iteration.

Number of epochs: Defines the number of times the entire training dataset is passed through the model.

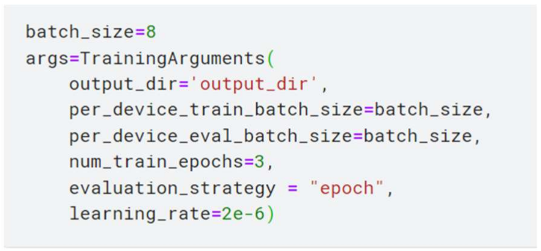
Outputs for Different Hyper-Parameters Tuning is Shown Below :-

For Indic-Bert Tuning :-

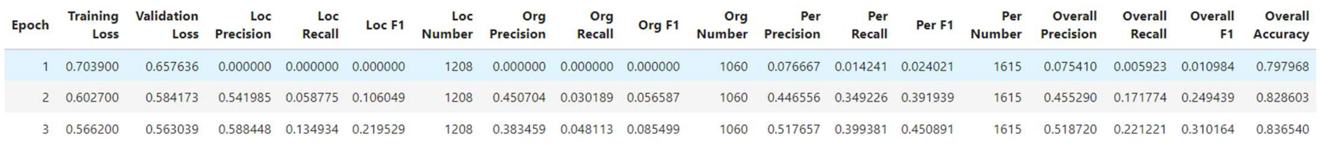
i.



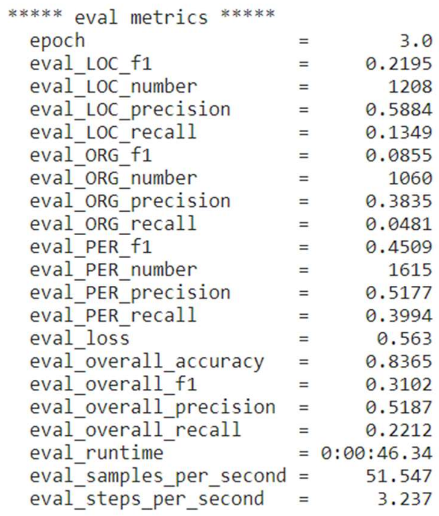
```
Precision   0.566970
Recall      0.250287
F1          0.347272
```



```
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)
```

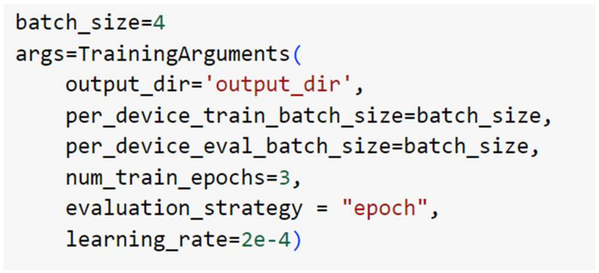


Epoch	Training Loss	Validation Loss	Loc Precision	Loc Recall	Loc F1	Loc Number	Org Precision	Org Recall	Org F1	Org Number	Per Precision	Per Recall	Per F1	Per Number	Overall Precision	Overall Recall	Overall F1	Overall Accuracy
1	0.703900	0.657636	0.000000	0.000000	0.000000	1208	0.000000	0.000000	0.000000	1060	0.076667	0.014241	0.024021	1615	0.075410	0.005923	0.010984	0.797968
2	0.602700	0.584173	0.541985	0.058775	0.106049	1208	0.450704	0.030189	0.056587	1060	0.446556	0.349226	0.391939	1615	0.455290	0.171774	0.249439	0.828603
3	0.566200	0.563039	0.588448	0.134934	0.219529	1208	0.383459	0.048113	0.085499	1060	0.517657	0.399381	0.450891	1615	0.518720	0.221221	0.310164	0.836540

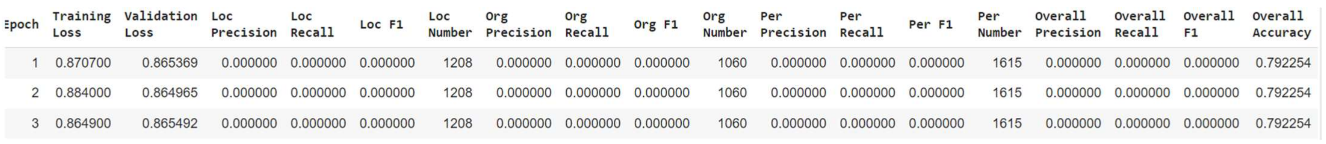


```
***** eval metrics *****
  epoch                   =        3.0
  eval_LOC_f1             =     0.2195
  eval_LOC_number         =       1208
  eval_LOC_precision      =     0.5884
  eval_LOC_recall         =     0.1349
  eval_ORG_f1             =     0.0855
  eval_ORG_number         =       1060
  eval_ORG_precision      =     0.3835
  eval_ORG_recall         =     0.0481
  eval_PER_f1             =     0.4509
  eval_PER_number         =       1615
  eval_PER_precision      =     0.5177
  eval_PER_recall         =     0.3994
  eval_loss               =      0.563
  eval_overall_accuracy   =     0.8365
  eval_overall_f1         =     0.3102
  eval_overall_precision  =     0.5187
  eval_overall_recall     =     0.2212
  eval_runtime            = 0:00:46.34
  eval_samples_per_second =     51.547
  eval_steps_per_second   =      3.237
```

ii.

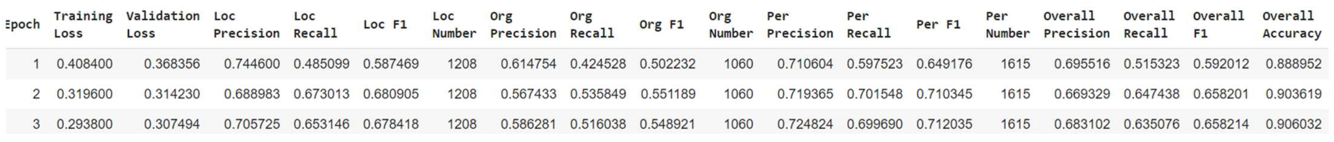


```
batch_size=4
args=TrainingArguments(
    output_dir='output_dir',
    per_device_train_batch_size=batch_size,
    per_device_eval_batch_size=batch_size,
    num_train_epochs=3,
    evaluation_strategy = "epoch",
    learning_rate=2e-4)
```

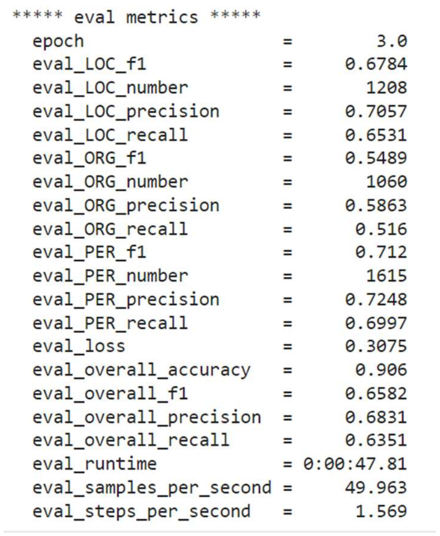


Epoch	Training Loss	Validation Loss	Loc Precision	Loc Recall	Loc F1	Loc Number	Org Precision	Org Recall	Org F1	Org Number	Per Precision	Per Recall	Per F1	Per Number	Overall Precision	Overall Recall	Overall F1	Overall Accuracy
1	0.870700	0.865369	0.000000	0.000000	0.000000	1208	0.000000	0.000000	0.000000	1060	0.000000	0.000000	0.000000	1615	0.000000	0.000000	0.000000	0.792254
2	0.884000	0.864965	0.000000	0.000000	0.000000	1208	0.000000	0.000000	0.000000	1060	0.000000	0.000000	0.000000	1615	0.000000	0.000000	0.000000	0.792254
3	0.864900	0.865492	0.000000	0.000000	0.000000	1208	0.000000	0.000000	0.000000	1060	0.000000	0.000000	0.000000	1615	0.000000	0.000000	0.000000	0.792254

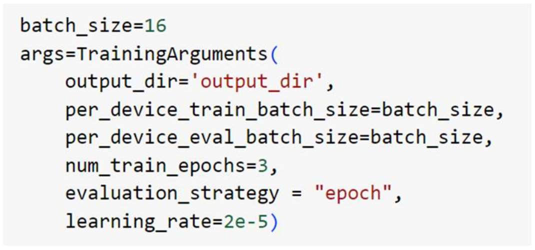
iii.



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	0.408400	0.368356	0.744600	0.485099	0.587469	1208	0.614754	0.424528	0.502232	1060	0.710604	0.597523	0.649176	1615	0.695516	0.515323	0.592012	0.888952
2	0.319600	0.314230	0.688983	0.673013	0.680905	1208	0.567433	0.535849	0.551189	1060	0.719365	0.701548	0.710345	1615	0.669329	0.647438	0.658201	0.903619
3	0.293800	0.307494	0.705725	0.653146	0.678418	1208	0.586281	0.516038	0.548921	1060	0.724824	0.699690	0.712035	1615	0.683102	0.635076	0.658214	0.906032



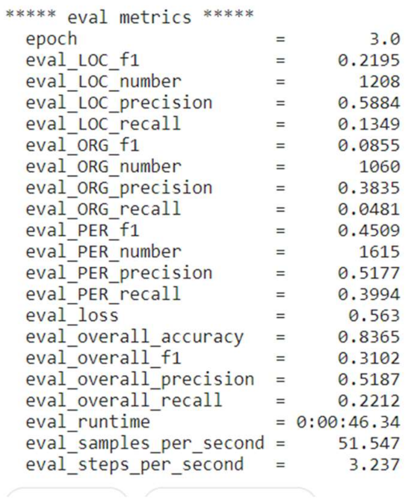
```
***** eval metrics *****
  epoch                   =        3.0
  eval_LOC_f1             =     0.6784
  eval_LOC_number         =       1208
  eval_LOC_precision      =     0.7057
  eval_LOC_recall         =     0.6531
  eval_ORG_f1             =     0.5489
  eval_ORG_number         =       1060
  eval_ORG_precision      =     0.5863
  eval_ORG_recall         =      0.516
  eval_PER_f1             =      0.712
  eval_PER_number         =       1615
  eval_PER_precision      =     0.7248
  eval_PER_recall         =     0.6997
  eval_loss               =     0.3075
  eval_overall_accuracy   =      0.906
  eval_overall_f1         =     0.6582
  eval_overall_precision  =     0.6831
  eval_overall_recall     =     0.6351
  eval_runtime            = 0:00:47.81
  eval_samples_per_second =     49.963
  eval_steps_per_second   =      1.569
```



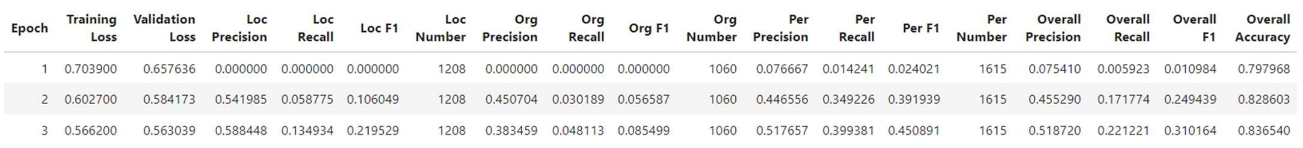
```
batch_size=16
args=TrainingArguments(
    output_dir='output_dir',
    per_device_train_batch_size=batch_size,
    per_device_eval_batch_size=batch_size,
    num_train_epochs=3,
    evaluation_strategy = "epoch",
    learning_rate=2e-5)
```

For Indic-Ner Tuning :-

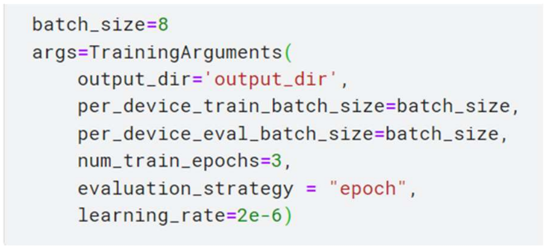
i.



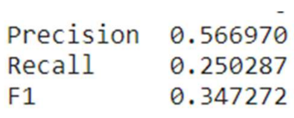
```
***** eval metrics *****
  epoch                   =        3.0
  eval_LOC_f1             =     0.2195
  eval_LOC_number         =       1208
  eval_LOC_precision      =     0.5884
  eval_LOC_recall         =     0.1349
  eval_ORG_f1             =     0.0855
  eval_ORG_number         =       1060
  eval_ORG_precision      =     0.3835
  eval_ORG_recall         =     0.0481
  eval_PER_f1             =     0.4509
  eval_PER_number         =       1615
  eval_PER_precision      =     0.5177
  eval_PER_recall         =     0.3994
  eval_loss               =      0.563
  eval_overall_accuracy   =     0.8365
  eval_overall_f1         =     0.3102
  eval_overall_precision  =     0.5187
  eval_overall_recall     =     0.2212
  eval_runtime            = 0:00:46.34
  eval_samples_per_second =     51.547
  eval_steps_per_second   =      3.237
```



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	0.703900	0.657636	0.000000	0.000000	0.000000	1208	0.000000	0.000000	0.000000	1060	0.076667	0.014241	0.024021	1615	0.075410	0.005923	0.010984	0.797968
2	0.602700	0.584173	0.541985	0.058775	0.106049	1208	0.450704	0.030189	0.056587	1060	0.446556	0.349226	0.391939	1615	0.455290	0.171774	0.249439	0.828603
3	0.566200	0.563039	0.588448	0.134934	0.219529	1208	0.383459	0.048113	0.085499	1060	0.517657	0.399381	0.450891	1615	0.518720	0.221221	0.310164	0.836540

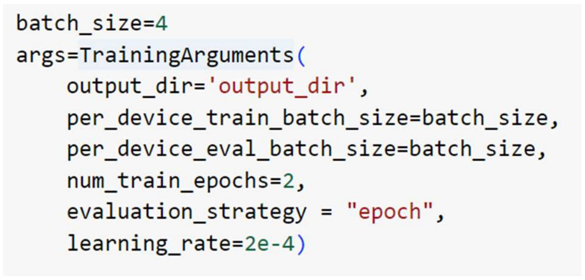


```
batch_size=8
args=TrainingArguments(
    output_dir='output_dir',
    per_device_train_batch_size=batch_size,
    per_device_eval_batch_size=batch_size,
    num_train_epochs=3,
    evaluation_strategy = "epoch",
    learning_rate=2e-6)
```

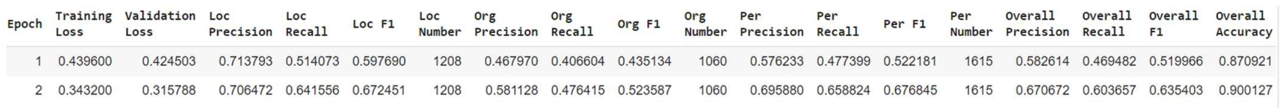


```
Precision   0.566970
Recall      0.250287
F1          0.347272
```

ii.

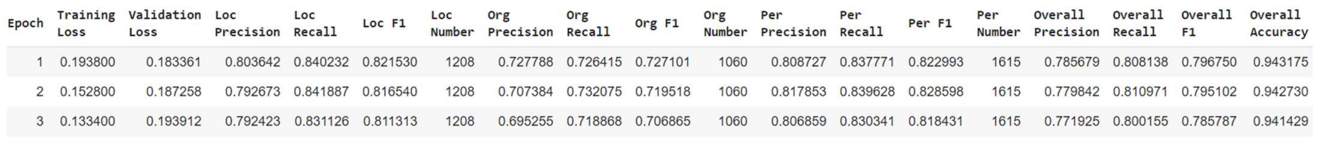


```
batch_size=4
args=TrainingArguments(
    output_dir='output_dir',
    per_device_train_batch_size=batch_size,
    per_device_eval_batch_size=batch_size,
    num_train_epochs=2,
    evaluation_strategy = "epoch",
    learning_rate=2e-4)
```

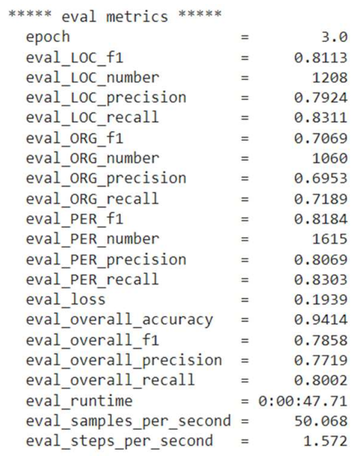


Epoch	Training Loss	Validation Loss	Loc Precision	Loc Recall	Loc F1	Loc Number	Org Precision	Org Recall	Org F1	Org Number	Per Precision	Per Recall	Per F1	Per Number	Overall Precision	Overall Recall	Overall F1	Overall Accuracy
1	0.439600	0.424503	0.713793	0.514073	0.597690	1208	0.467970	0.406604	0.435134	1060	0.576233	0.477399	0.522181	1615	0.582614	0.469482	0.519966	0.870921
2	0.343200	0.315788	0.706472	0.641556	0.672451	1208	0.581128	0.476415	0.523587	1060	0.695880	0.658824	0.676845	1615	0.670672	0.603657	0.635403	0.900127

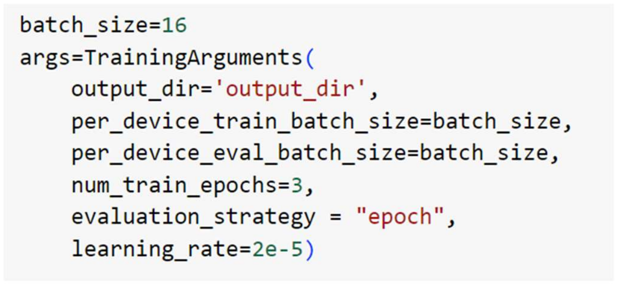
iii.



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	0.193800	0.183361	0.803642	0.840232	0.821530	1208	0.727788	0.726415	0.727101	1060	0.808727	0.837771	0.822993	1615	0.785679	0.808138	0.796750	0.943175
2	0.152800	0.187258	0.792673	0.841887	0.816540	1208	0.707384	0.732075	0.719518	1060	0.817853	0.839628	0.828598	1615	0.779842	0.810971	0.795102	0.942730
3	0.133400	0.193912	0.792423	0.831126	0.811313	1208	0.695255	0.718868	0.706865	1060	0.806859	0.830341	0.818431	1615	0.771925	0.800155	0.785787	0.941429



```
***** eval metrics *****
  epoch                   =        3.0
  eval_LOC_f1             =     0.8113
  eval_LOC_number         =       1208
  eval_LOC_precision      =     0.7924
  eval_LOC_recall         =     0.8311
  eval_ORG_f1             =     0.7069
  eval_ORG_number         =       1060
  eval_ORG_precision      =     0.6953
  eval_ORG_recall         =     0.7189
  eval_PER_f1             =     0.8184
  eval_PER_number         =       1615
  eval_PER_precision      =     0.8069
  eval_PER_recall         =     0.8303
  eval_loss               =     0.1939
  eval_overall_accuracy   =     0.9414
  eval_overall_f1         =     0.7858
  eval_overall_precision  =     0.7719
  eval_overall_recall     =     0.8002
  eval_runtime            = 0:00:47.71
  eval_samples_per_second =     50.068
  eval_steps_per_second   =      1.572
```



```
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=2e-5)
```

4. Optimal Values

After experimentation and validation on the validation set, the optimal values chosen for the hyper-parameters were as follows:

For Indic-Bert :-

Learning rate: 2e-5

Batch size: 16

Number of epochs: 3

For Indic-Ner :-

Learning rate: 2e-5

Batch size: 16

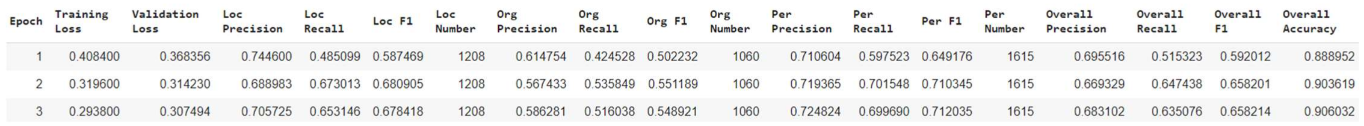
Number of epochs: 3

5. Model Outputs

{Consider that I’ve run Models on 40000 lines. For 20% data it takes more than 3.5 hours and for 40000 lines it takes 1.5 hours to train the model}

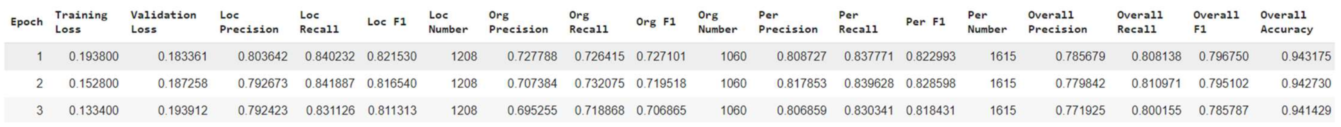
The outputs are shown below:-

Indic-Bert Training Outputs :-



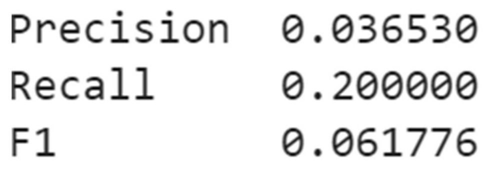
Epoch	Training Loss	Validation Loss	Loc Precision	Loc Recall	Loc F1	Loc Number	Org Precision	Org Recall	Org F1	Org Number	Per Precision	Per Recall	Per F1	Per Number	Overall Precision	Overall Recall	Overall F1	Overall Accuracy
1	0.408400	0.368356	0.744600	0.485099	0.587469	1208	0.614754	0.424528	0.502232	1060	0.710604	0.597523	0.649176	1615	0.695516	0.515323	0.592012	0.888952
2	0.319600	0.314230	0.688983	0.673013	0.680905	1208	0.567433	0.535849	0.551189	1060	0.719365	0.701548	0.710345	1615	0.669329	0.647438	0.658201	0.903619
3	0.293800	0.307494	0.705725	0.653146	0.678418	1208	0.586281	0.516038	0.548921	1060	0.724824	0.699690	0.712035	1615	0.683102	0.635076	0.658214	0.906032

Indic-Ner Training Outputs :-



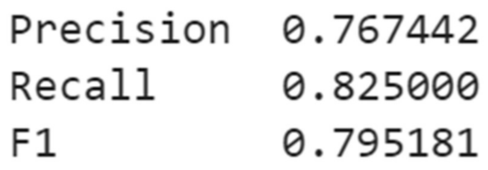
Epoch	Training Loss	Validation Loss	Loc Precision	Loc Recall	Loc F1	Loc Number	Org Precision	Org Recall	Org F1	Org Number	Per Precision	Per Recall	Per F1	Per Number	Overall Precision	Overall Recall	Overall F1	Overall Accuracy
1	0.193800	0.183361	0.803642	0.840232	0.821530	1208	0.727788	0.726415	0.727101	1060	0.808727	0.837771	0.822993	1615	0.785679	0.808138	0.796750	0.943175
2	0.152800	0.187258	0.792673	0.841887	0.816540	1208	0.707384	0.732075	0.719518	1060	0.817853	0.839628	0.828598	1615	0.779842	0.810971	0.795102	0.942730
3	0.133400	0.193912	0.792423	0.831126	0.811313	1208	0.695255	0.718868	0.706865	1060	0.806859	0.830341	0.818431	1615	0.771925	0.800155	0.785787	0.941429

Indic-BERT Test Performance :-



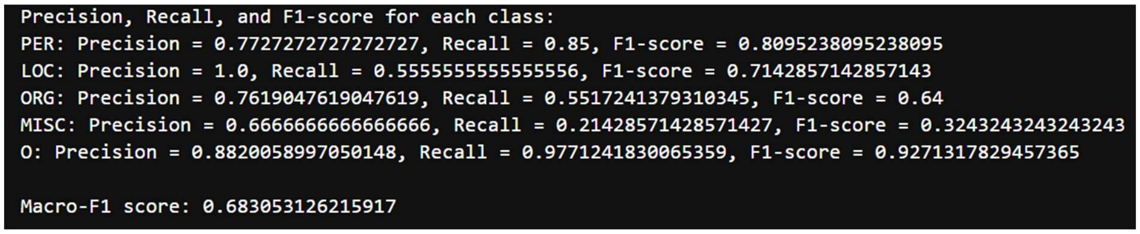
```
Precision  0.036530
Recall     0.200000
F1         0.061776
```

Indic-Ner Test Performance :-



```
Precision  0.767442
Recall     0.825000
F1         0.795181
```

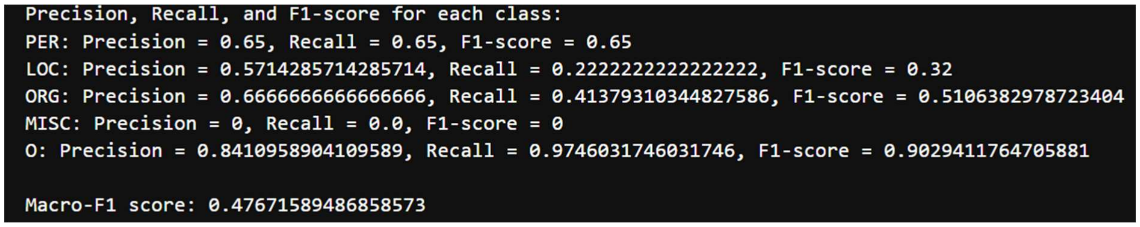
Comparison of Manually Outputs vs ChatGPT Outputs for 25 Sentences :-



```
Precision, Recall, and F1-score for each class:
PER: Precision = 0.7727272727272727, Recall = 0.85, F1-score = 0.8095238095238095
LOC: Precision = 1.0, Recall = 0.5555555555555556, F1-score = 0.7142857142857143
ORG: Precision = 0.7619047619047619, Recall = 0.5517241379310345, F1-score = 0.64
MISC: Precision = 0.6666666666666666, Recall = 0.21428571428571427, F1-score = 0.3243243243243243
O: Precision = 0.8820058997050148, Recall = 0.9771241830065359, F1-score = 0.9271317829457365

Macro-F1 score: 0.683053126215917
```

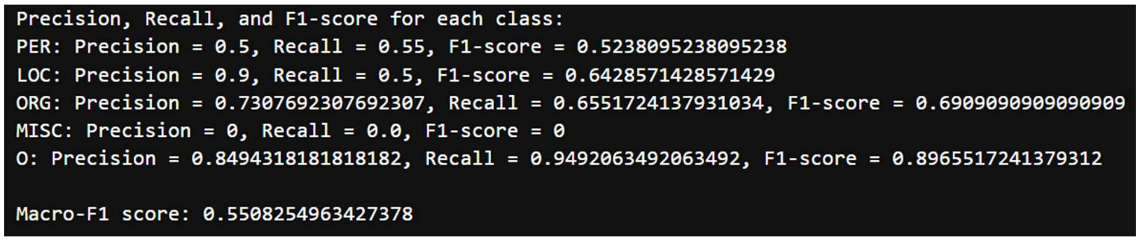
Comparison of Manually Outputs vs Indic-Bert Outputs for 25 Sentences :-



```
Precision, Recall, and F1-score for each class:
PER: Precision = 0.65, Recall = 0.65, F1-score = 0.65
LOC: Precision = 0.5714285714285714, Recall = 0.2222222222222222, F1-score = 0.32
ORG: Precision = 0.6666666666666666, Recall = 0.41379310344827586, F1-score = 0.5106382978723404
MISC: Precision = 0, Recall = 0.0, F1-score = 0
O: Precision = 0.8410958904109589, Recall = 0.9746031746031746, F1-score = 0.9029411764705881

Macro-F1 score: 0.47671589486858573
```

Comparison of Manually Outputs vs Indic-Ner Outputs for 25 Sentences :-



```
Precision, Recall, and F1-score for each class:
PER: Precision = 0.5, Recall = 0.55, F1-score = 0.5238095238095238
LOC: Precision = 0.9, Recall = 0.5, F1-score = 0.6428571428571429
ORG: Precision = 0.7307692307692307, Recall = 0.6551724137931034, F1-score = 0.6909090909090909
MISC: Precision = 0, Recall = 0.0, F1-score = 0
O: Precision = 0.8494318181818182, Recall = 0.9492063492063492, F1-score = 0.8965517241379312

Macro-F1 score: 0.5508254963427378
```