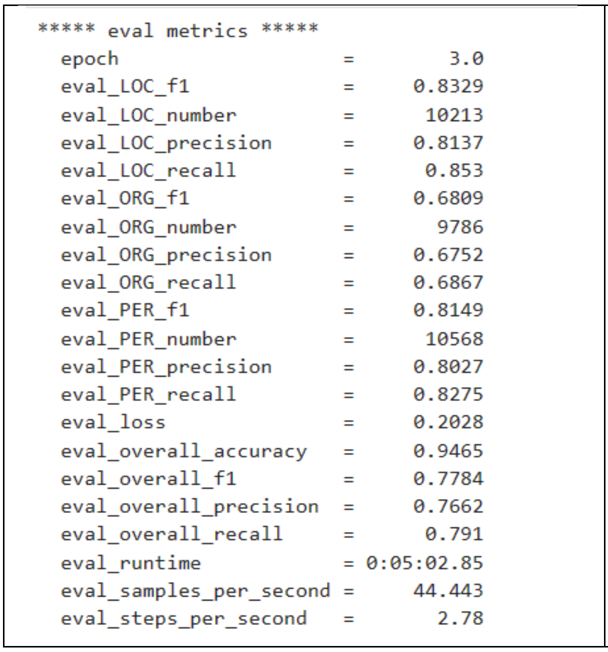
Comparison of both Model on Namapadam Corpus(Hindi Language)

We train both the model using 20000 sentences of training data of Namapadam corpus and evaluate the performance of both models on validation and test data set.

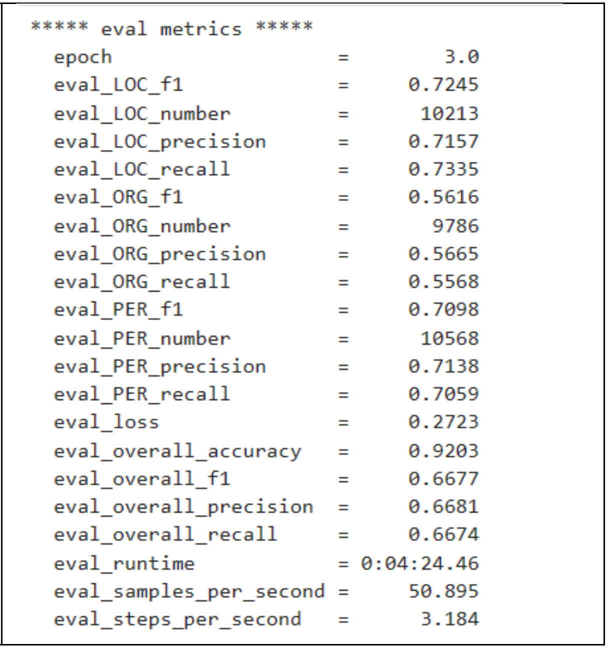
As we can see in the below figure the Overall F1 score(macro f1 score) on validation data set using Indic NER and IndicBERT

IndicNER



```
***** eval metrics *****
  epoch                   =        3.0
  eval_LOC_f1             =     0.8329
  eval_LOC_number         =      10213
  eval_LOC_precision      =     0.8137
  eval_LOC_recall         =      0.853
  eval_ORG_f1             =     0.6809
  eval_ORG_number         =       9786
  eval_ORG_precision      =     0.6752
  eval_ORG_recall         =     0.6867
  eval_PER_f1             =     0.8149
  eval_PER_number         =      10568
  eval_PER_precision      =     0.8027
  eval_PER_recall         =     0.8275
  eval_loss               =     0.2028
  eval_overall_accuracy   =     0.9465
  eval_overall_f1         =     0.7784
  eval_overall_precision  =     0.7662
  eval_overall_recall     =      0.791
  eval_runtime            = 0:05:02.85
  eval_samples_per_second =     44.443
  eval_steps_per_second   =       2.78
```

IndicBERT

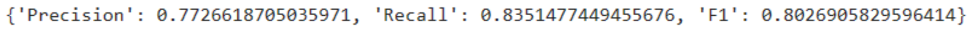


```
***** eval metrics *****
  epoch                   =        3.0
  eval_LOC_f1             =     0.7245
  eval_LOC_number         =      10213
  eval_LOC_precision      =     0.7157
  eval_LOC_recall         =     0.7335
  eval_ORG_f1             =     0.5616
  eval_ORG_number         =       9786
  eval_ORG_precision      =     0.5665
  eval_ORG_recall         =     0.5568
  eval_PER_f1             =     0.7098
  eval_PER_number         =      10568
  eval_PER_precision      =     0.7138
  eval_PER_recall         =     0.7059
  eval_loss               =     0.2723
  eval_overall_accuracy   =     0.9203
  eval_overall_f1         =     0.6677
  eval_overall_precision  =     0.6681
  eval_overall_recall     =     0.6674
  eval_runtime            = 0:04:24.46
  eval_samples_per_second =     50.895
  eval_steps_per_second   =      3.184
```

As we can see that IndicNER has better overall accuracy and better overall f1 score(macro f1 score) then IndicBERT here IndicNER performs better.

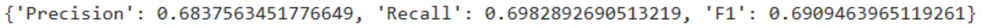
As we can see in the below figure the Overall F1 score(macro f1 score) on test data set using Indic NER and IndicBERT

IndicNER:



```
{'Precision': 0.7726618705035971, 'Recall': 0.8351477449455676, 'F1': 0.8026905829596414}
```

IndicBERT:

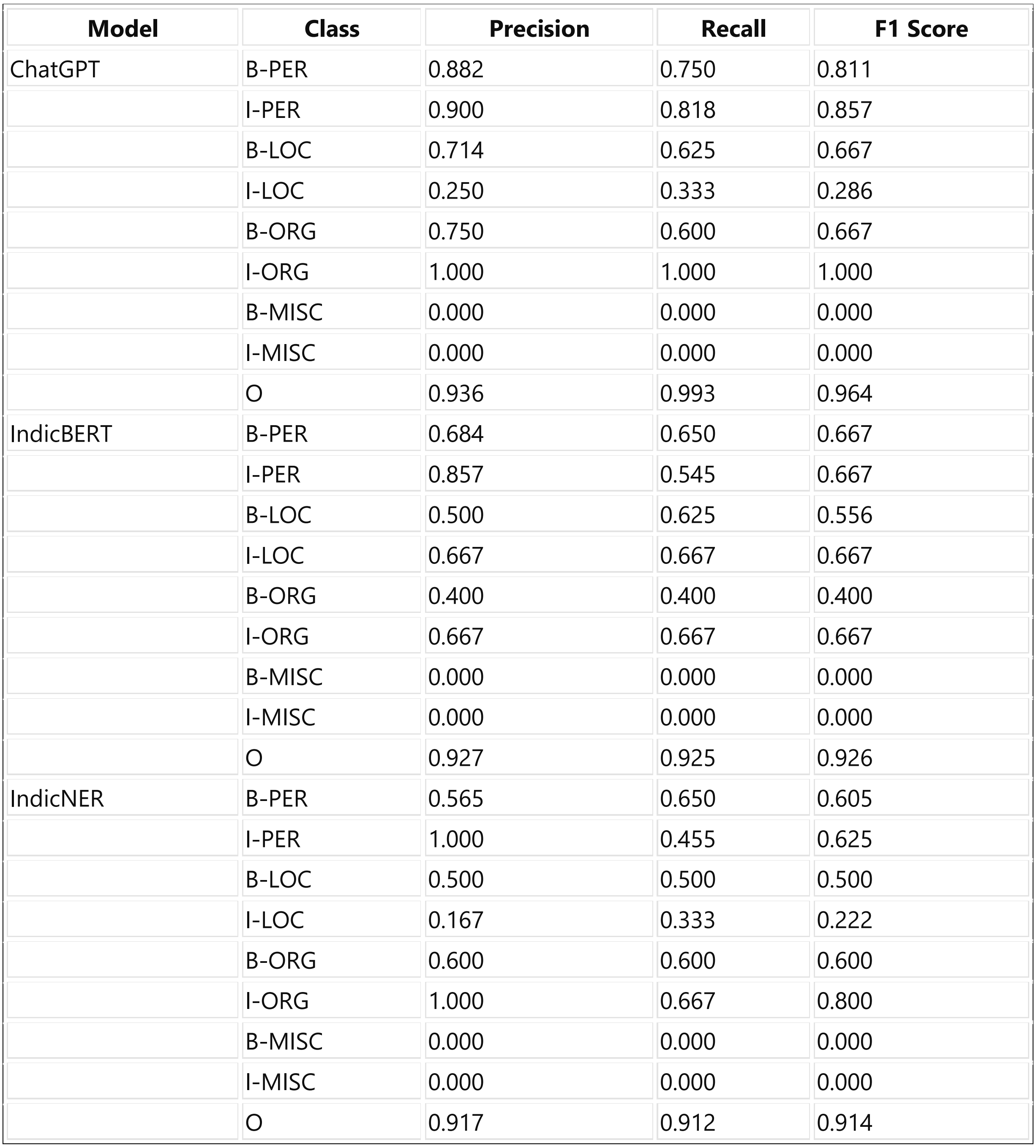


```
{'Precision': 0.6837563451776649, 'Recall': 0.6982892690513219, 'F1': 0.6909463965119261}
```

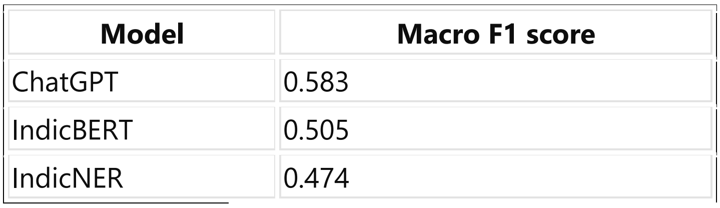
As we can see that IndicNER has better overall accuracy and better overall f1 score(macro f1 score) then IndicBERT here IndicNER performs better.

So Overall we can say that for NER task IndicNER perfoms better then IndicBERT on Namapadam Corpus.

Comparison of both Model and ChatGPT on 25 Sentences



Model	Class	Precision	Recall	F1 Score
ChatGPT	B-PER	0.882	0.750	0.811
	I-PER	0.900	0.818	0.857
	B-LOC	0.714	0.625	0.667
	I-LOC	0.250	0.333	0.286
	B-ORG	0.750	0.600	0.667
	I-ORG	1.000	1.000	1.000
	B-MISC	0.000	0.000	0.000
	I-MISC	0.000	0.000	0.000
	O	0.936	0.993	0.964
IndicBERT	B-PER	0.684	0.650	0.667
	I-PER	0.857	0.545	0.667
	B-LOC	0.500	0.625	0.556
	I-LOC	0.667	0.667	0.667
	B-ORG	0.400	0.400	0.400
	I-ORG	0.667	0.667	0.667
	B-MISC	0.000	0.000	0.000
	I-MISC	0.000	0.000	0.000
	O	0.927	0.925	0.926
IndicNER	B-PER	0.565	0.650	0.605
	I-PER	1.000	0.455	0.625
	B-LOC	0.500	0.500	0.500
	I-LOC	0.167	0.333	0.222
	B-ORG	0.600	0.600	0.600
	I-ORG	1.000	0.667	0.800
	B-MISC	0.000	0.000	0.000
	I-MISC	0.000	0.000	0.000
	O	0.917	0.912	0.914



Model	Macro F1 score
ChatGPT	0.583
IndicBERT	0.505
IndicNER	0.474

As we can see that ChatGPT leads in the overall F1 score, followed by IndicBERT and IndicNER.

Learning from this comparison :

1. In case of Namapadam Corpus IndicNER, designed specifically for Named Entity Recognition (NER) tasks, may hold an inherent advantage over IndicBERT when it comes to handling NER tasks seamlessly. This could potentially explain why IndicNER demonstrates fewer errors or achieves a higher overall F1 score compared to Indic-BERT as we can see in Q2. Indic-BERT, while being a versatile language model fine-tuned for various tasks including NER, might not exhibit the same level of performance as Indic-NER

2. While on 25 Sentences here ChatGPT performs better beacsue we have trained IndicBERT and IndicNER with 20000 sentences which is very less so IndicBERT and IndicNER are not performing better compared to chatGPT.As you can see we have trained IndicBERT with 20% of training data (approx 1.5 lakhs sentences) then it is performing almost as chatGPT(see Q4).

3. Here with same amount of training IndicNER is better then IndicBERT in case of Namapadam corpus but on 25 sentences IndicBERT performs better but there is just a small difference we can observe.So for more sentences we may not observe that.

4. IndicNER has very less change when we change hyperparameters.

FineTuning of IndicBERT and IndicNER for NER Task

Hyperparameters plays a crucial role in training the model and can significantly impact its performance and convergence. I have choosed following hyperparameters and their significance and optimal values as follows::

1. **per_device_train_batch_size** ::This hyperparameter determines the batch size of training samples per device (e.g., GPU) during training. Larger batch sizes can lead to faster training but may require more memory. Smaller batch sizes may provide better generalization but slower convergence.

Optimal Value for Both IndicBERT and IndicNER is 8

2. **per_device_eval_batch_size::**Similar to per_device_train_batch_size but for evaluation/validation data. It determines the batch size of evaluation samples per device during evaluation.

Optimal Value for Both IndicBERT and IndicNER is 8

3. **num_train_epochs::**This hyperparameter specifies the number of times the entire training dataset is passed through the model during training. Increasing the number of epochs may improve model performance, but it could also lead to overfitting if the model learns to memorize the training data.

Optimal Value for Both IndicBERT and IndicNER is 3

4. **evaluation_strategy::**This hyperparameter determines when evaluation is performed during training. "epoch" means evaluation is performed at the end of each epoch. Other options could include steps or no evaluation during training.

I have choosed evaluation strategy is epoch. I have not changed/Tuned this.

5. **learning_rate:**This hyperparameter determines the step size at which the model weights are updated during training. A higher learning rate may lead to faster convergence but could cause instability or overshooting. A lower learning rate may lead to slower convergence but more stable training.

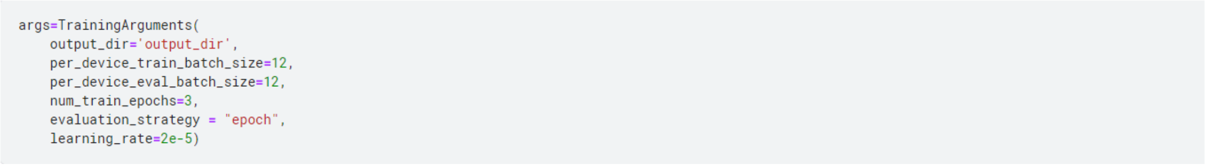
Optimal Value for Both IndicBERT and IndicNER is 4e-5

6. **weight_decay::**Weight decay is a regularization technique that penalizes large weights in the model during training to prevent overfitting. It reduces the magnitude of the weights during optimization, effectively adding a penalty term to the loss function.

Optimal Value for Both IndicBERT and IndicNER is 0.01

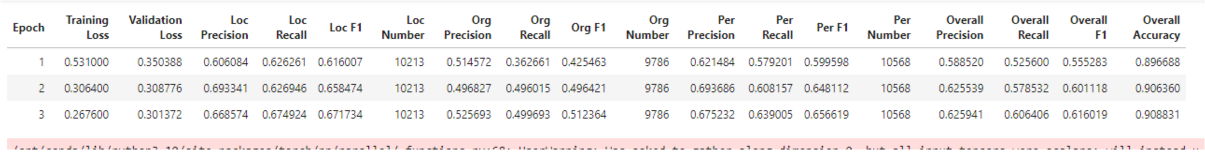
Finetuning of IndicBERT

1.



```
args=TrainingArguments(
    output_dir='output_dir',
    per_device_train_batch_size=12,
    per_device_eval_batch_size=12,
    num_train_epochs=3,
    evaluation_strategy = "epoch",
    learning_rate=2e-5)
```

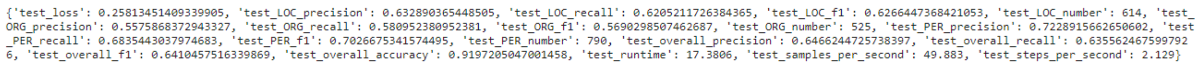
Output on Validation Data set:



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	0.531000	0.350388	0.606084	0.626261	0.616007	10213	0.514572	0.362661	0.425463	9786	0.621484	0.579201	0.599598	10568	0.588520	0.525600	0.555283	0.896688
2	0.306400	0.308776	0.693341	0.626946	0.658474	10213	0.496827	0.496015	0.496421	9786	0.693686	0.608157	0.648112	10568	0.625539	0.578532	0.601118	0.906360
3	0.267600	0.301372	0.668574	0.674924	0.671734	10213	0.525693	0.499693	0.512364	9786	0.675232	0.639005	0.656619	10568	0.625941	0.606406	0.616019	0.908831

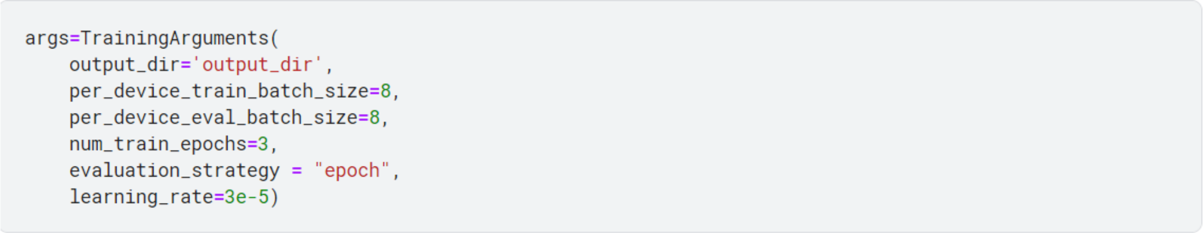
```
/opt/conda/lib/python3.10/site-packages/torch/nn/parallel/_functions.py:68: UserWarning: Was asked to gather along dimension 0, but all input tensors were scalars; will instead u
```

Output on Test Data set:



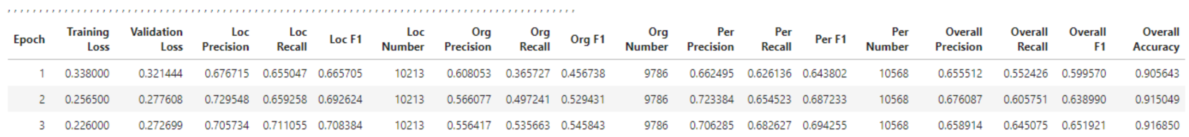
```
{'test_loss': 0.25813451409339905, 'test_LOC_precision': 0.632890365448505, 'test_LOC_recall': 0.6205211726384365, 'test_LOC_f1': 0.6266447368421053, 'test_LOC_number': 614, 'test_
ORG_precision': 0.5575868372943327, 'test_ORG_recall': 0.580952380952381, 'test_ORG_f1': 0.5690298507462687, 'test_ORG_number': 525, 'test_PER_precision': 0.7228915662650602, 'test
_PER_recall': 0.6835443037974683, 'test_PER_f1': 0.7026675341574495, 'test_PER_number': 790, 'test_overall_precision': 0.6466244725738397, 'test_overall_recall': 0.635562467599792
6, 'test_overall_f1': 0.6410457516339869, 'test_overall_accuracy': 0.9197205047001458, 'test_runtime': 17.3806, 'test_samples_per_second': 49.883, 'test_steps_per_second': 2.129}
```

2.



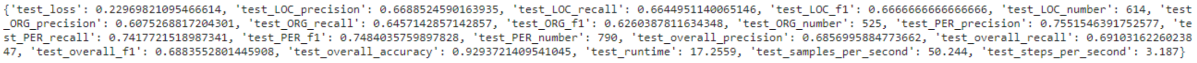
```
args=TrainingArguments(
    output_dir='output_dir',
    per_device_train_batch_size=8,
    per_device_eval_batch_size=8,
    num_train_epochs=3,
    evaluation_strategy = "epoch",
    learning_rate=3e-5)
```

Output on Validation Data set:



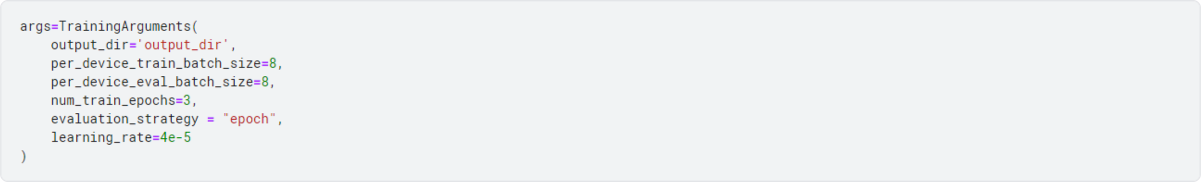
Epoch	Training Loss	Validation Loss	Loc Precision	Loc Recall	Loc F1	Loc Number	Org Precision	Org Recall	Org F1	Org Number	Per Precision	Per Recall	Per F1	Per Number	Overall Precision	Overall Recall	Overall F1	Overall Accuracy
1	0.338000	0.321444	0.676715	0.655047	0.665705	10213	0.608053	0.365727	0.456738	9786	0.662495	0.626136	0.643802	10568	0.655512	0.552426	0.599570	0.905643
2	0.256500	0.277608	0.729548	0.659258	0.692624	10213	0.566077	0.497241	0.529431	9786	0.723384	0.654523	0.687233	10568	0.676087	0.605751	0.638990	0.915049
3	0.226000	0.272699	0.705734	0.711055	0.708384	10213	0.556417	0.535663	0.545843	9786	0.706285	0.682627	0.694255	10568	0.658914	0.645075	0.651921	0.916850

Output on Test Data set:



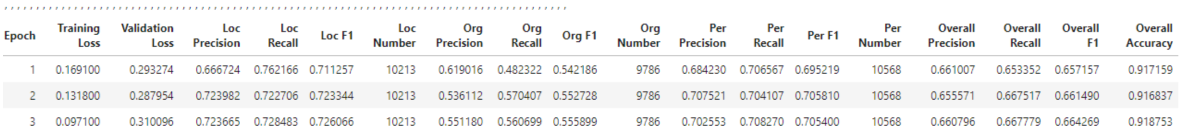
```
{'test_loss': 0.22969821095466614, 'test_LOC_precision': 0.6688524590163935, 'test_LOC_recall': 0.6644951140065146, 'test_LOC_f1': 0.6666666666666666, 'test_LOC_number': 614, 'test
_ORG_precision': 0.6075268817204301, 'test_ORG_recall': 0.6457142857142857, 'test_ORG_f1': 0.6260387811634348, 'test_ORG_number': 525, 'test_PER_precision': 0.7551546391752577, 'te
st_PER_recall': 0.7417721518987341, 'test_PER_f1': 0.7484035759897828, 'test_PER_number': 790, 'test_overall_precision': 0.6856995884773662, 'test_overall_recall': 0.69103162260238
47, 'test_overall_f1': 0.6883552801445908, 'test_overall_accuracy': 0.9293721409541045, 'test_runtime': 17.2559, 'test_samples_per_second': 50.244, 'test_steps_per_second': 3.187}
```

3.



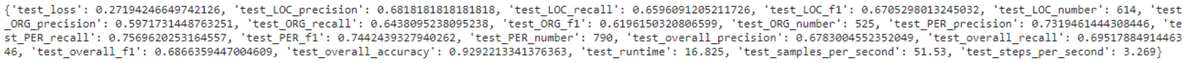
```
args=TrainingArguments(
    output_dir='output_dir',
    per_device_train_batch_size=8,
    per_device_eval_batch_size=8,
    num_train_epochs=3,
    evaluation_strategy = "epoch",
    learning_rate=4e-5
)
```

Output on Validation Data set:



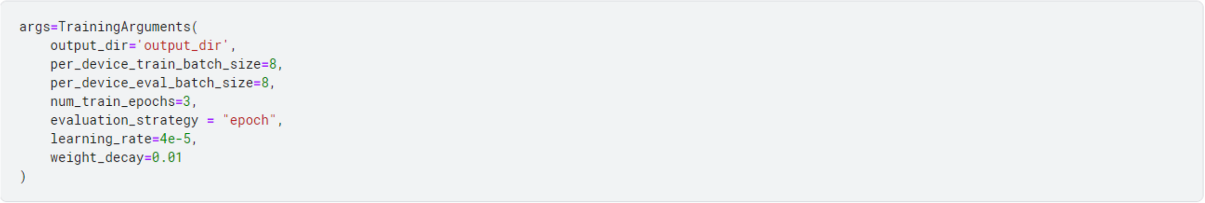
Epoch	Training Loss	Validation Loss	Loc Precision	Loc Recall	Loc F1	Loc Number	Org Precision	Org Recall	Org F1	Org Number	Per Precision	Per Recall	Per F1	Per Number	Overall Precision	Overall Recall	Overall F1	Overall Accuracy
1	0.169100	0.293274	0.666724	0.762166	0.711257	10213	0.619016	0.482322	0.542186	9786	0.684230	0.706567	0.695219	10568	0.661007	0.653352	0.657157	0.917159
2	0.131800	0.287954	0.723982	0.722706	0.723344	10213	0.536112	0.570407	0.552728	9786	0.707521	0.704107	0.705810	10568	0.655571	0.667517	0.661490	0.916837
3	0.097100	0.310096	0.723665	0.728483	0.726066	10213	0.551180	0.560699	0.555899	9786	0.702553	0.708270	0.705400	10568	0.660796	0.667779	0.664269	0.918753

Output on Test Data set:



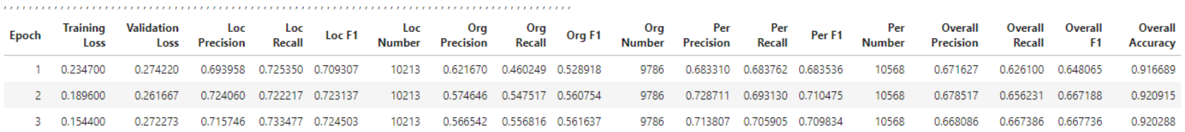
```
{'test_loss': 0.27194246649742126, 'test_LOC_precision': 0.6818181818181818, 'test_LOC_recall': 0.6596091205211726, 'test_LOC_f1': 0.6705298013245032, 'test_LOC_number': 614, 'test
_ORG_precision': 0.5971731448763251, 'test_ORG_recall': 0.6438095238095238, 'test_ORG_f1': 0.6196150320806599, 'test_ORG_number': 525, 'test_PER_precision': 0.7319461444308446, 'te
st_PER_recall': 0.7569620253164557, 'test_PER_f1': 0.7442439327940262, 'test_PER_number': 790, 'test_overall_precision': 0.6783004552352049, 'test_overall_recall': 0.69517884914463
46, 'test_overall_f1': 0.6866359447004609, 'test_overall_accuracy': 0.9292213341376363, 'test_runtime': 16.825, 'test_samples_per_second': 51.53, 'test_steps_per_second': 3.269}
```

4.



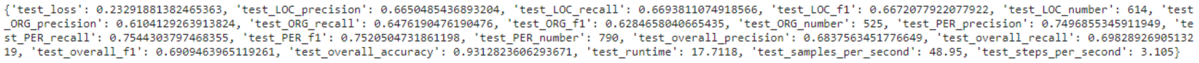
```
args=TrainingArguments(
    output_dir='output_dir',
    per_device_train_batch_size=8,
    per_device_eval_batch_size=8,
    num_train_epochs=3,
    evaluation_strategy = "epoch",
    learning_rate=4e-5,
    weight_decay=0.01
)
```

Output on Validation Data set:



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	0.234700	0.274220	0.693958	0.725350	0.709307	10213	0.621670	0.460249	0.528918	9786	0.683310	0.683762	0.683536	10568	0.671627	0.626100	0.648065	0.916689
2	0.189600	0.261667	0.724060	0.722217	0.723137	10213	0.574646	0.547517	0.560754	9786	0.728711	0.693130	0.710475	10568	0.678517	0.656231	0.667188	0.920915
3	0.154400	0.272273	0.715746	0.733477	0.724503	10213	0.566542	0.556816	0.561637	9786	0.713807	0.705905	0.709834	10568	0.668086	0.667386	0.667736	0.920288

Output on Test Data set:



```
{'test_loss': 0.23291881382465363, 'test_LOC_precision': 0.6650485436893204, 'test_LOC_recall': 0.6693811074918566, 'test_LOC_f1': 0.6672077922077922, 'test_LOC_number': 614, 'test
_ORG_precision': 0.6104129263913824, 'test_ORG_recall': 0.6476190476190476, 'test_ORG_f1': 0.6284658040665435, 'test_ORG_number': 525, 'test_PER_precision': 0.7496855345911949, 'te
st_PER_recall': 0.7544303797468355, 'test_PER_f1': 0.7520504731861198, 'test_PER_number': 790, 'test_overall_precision': 0.6837563451776649, 'test_overall_recall': 0.69828926905132
19, 'test_overall_f1': 0.6909463965119261, 'test_overall_accuracy': 0.9312823606293671, 'test_runtime': 17.7118, 'test_samples_per_second': 48.95, 'test_steps_per_second': 3.105}
```

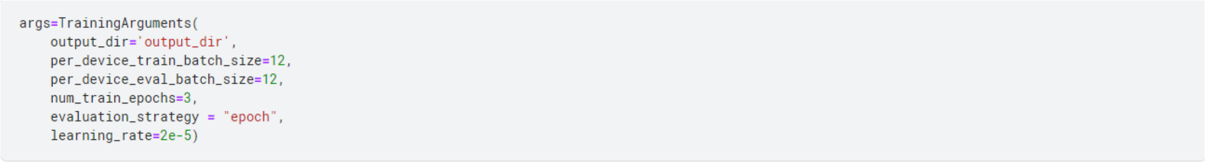
Conclusion:

As we can see that batch size =8 gives best performance and after adjust the learning rate and adding weight decay gives me best result as you can see in 4th iteration

As we can see that on 4th iteration we get Best Overall f1 score/macro f1 score on validation data set and test data set as well same of overall accuracy as well .

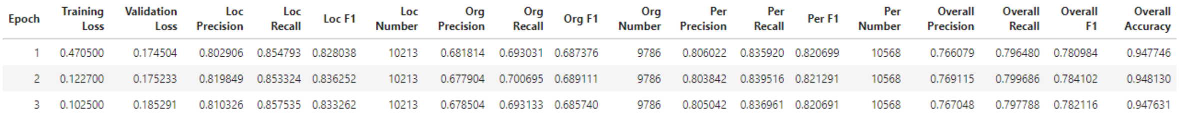
Finetuning of IndicNER

1.



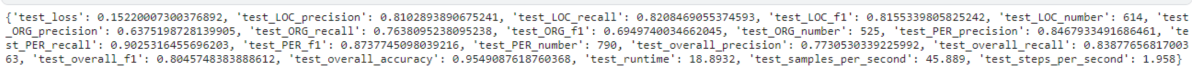
```
args=TrainingArguments(
    output_dir='output_dir',
    per_device_train_batch_size=12,
    per_device_eval_batch_size=12,
    num_train_epochs=3,
    evaluation_strategy = "epoch",
    learning_rate=2e-5)
```

Output on Validation Data set:



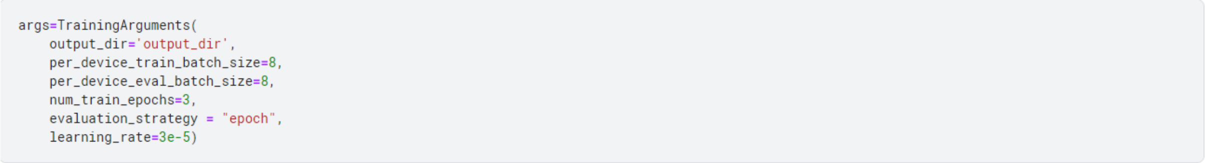
Epoch	Training Loss	Validation Loss	Loc Precision	Loc Recall	Loc F1	Loc Number	Org Precision	Org Recall	Org F1	Org Number	Per Precision	Per Recall	Per F1	Per Number	Overall Precision	Overall Recall	Overall F1	Overall Accuracy
1	0.470500	0.174504	0.802906	0.854793	0.828038	10213	0.681814	0.693031	0.687376	9786	0.806022	0.835920	0.820699	10568	0.766079	0.796480	0.780984	0.947746
2	0.122700	0.175233	0.819849	0.853324	0.836252	10213	0.677904	0.700695	0.689111	9786	0.803842	0.839516	0.821291	10568	0.769115	0.799686	0.784102	0.948130
3	0.102500	0.185291	0.810326	0.857535	0.833262	10213	0.678504	0.693133	0.685740	9786	0.805042	0.836961	0.820691	10568	0.767048	0.797788	0.782116	0.947631

Output on Test Data set:



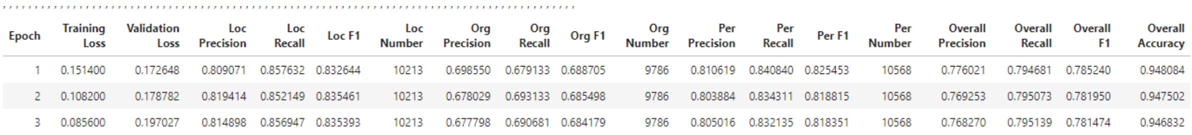
```
{'test_loss': 0.15220007300376892, 'test_LOC_precision': 0.8102893890675241, 'test_LOC_recall': 0.8208469055374593, 'test_LOC_f1': 0.8155339805825242, 'test_LOC_number': 614, 'test
_ORG_precision': 0.6375198728139905, 'test_ORG_recall': 0.7638095238095238, 'test_ORG_f1': 0.6949740034662045, 'test_ORG_number': 525, 'test_PER_precision': 0.8467933491686461, 'te
st_PER_recall': 0.9025316455696203, 'test_PER_f1': 0.8737745098039216, 'test_PER_number': 790, 'test_overall_precision': 0.7730530339225992, 'test_overall_recall': 0.83877656817003
63, 'test_overall_f1': 0.8045748383888612, 'test_overall_accuracy': 0.9549087618760368, 'test_runtime': 18.8932, 'test_samples_per_second': 45.889, 'test_steps_per_second': 1.958}
```

2.



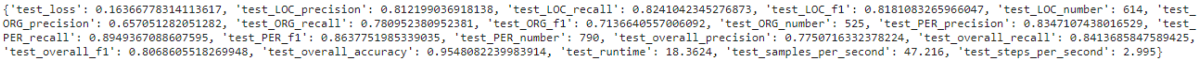
```
args=TrainingArguments(
    output_dir='output_dir',
    per_device_train_batch_size=8,
    per_device_eval_batch_size=8,
    num_train_epochs=3,
    evaluation_strategy = "epoch",
    learning_rate=3e-5)
```

Output on Validation Data set:



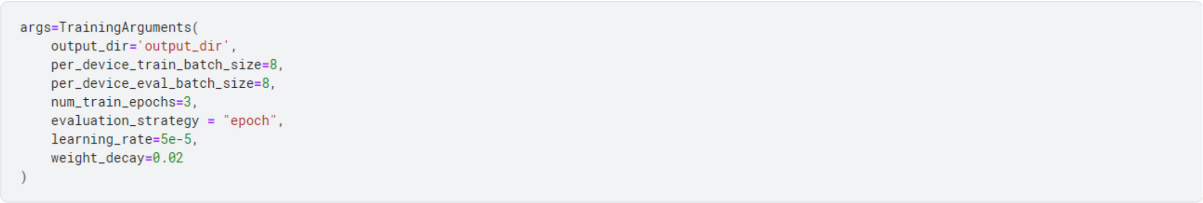
Epoch	Training Loss	Validation Loss	Loc Precision	Loc Recall	Loc F1	Loc Number	Org Precision	Org Recall	Org F1	Org Number	Per Precision	Per Recall	Per F1	Per Number	Overall Precision	Overall Recall	Overall F1	Overall Accuracy
1	0.151400	0.172648	0.809071	0.857632	0.832644	10213	0.698550	0.679133	0.688705	9786	0.810619	0.840840	0.825453	10568	0.776021	0.794681	0.785240	0.948084
2	0.108200	0.178782	0.819414	0.852149	0.835461	10213	0.678029	0.693133	0.685498	9786	0.803884	0.834311	0.818815	10568	0.769253	0.795073	0.781950	0.947502
3	0.085600	0.197027	0.814898	0.856947	0.835393	10213	0.677798	0.690681	0.684179	9786	0.805016	0.832135	0.818351	10568	0.768270	0.795139	0.781474	0.946832

Output on Test Data set:



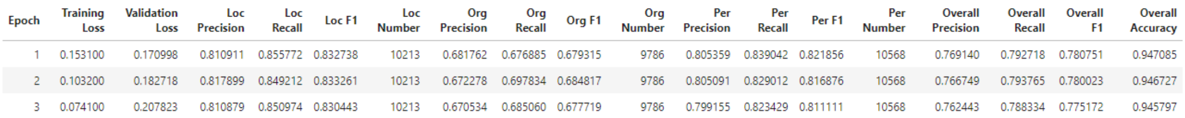
```
{'test_loss': 0.16366778314113617, 'test_LOC_precision': 0.812199036918138, 'test_LOC_recall': 0.8241042345276873, 'test_LOC_f1': 0.8181083265966047, 'test_LOC_number': 614, 'test_
ORG_precision': 0.657051282051282, 'test_ORG_recall': 0.780952380952381, 'test_ORG_f1': 0.7136640557006092, 'test_ORG_number': 525, 'test_PER_precision': 0.8347107438016529, 'test_
PER_recall': 0.8949367088607595, 'test_PER_f1': 0.8637751985339035, 'test_PER_number': 790, 'test_overall_precision': 0.7750716332378224, 'test_overall_recall': 0.8413685847589425,
'test_overall_f1': 0.8068605518269948, 'test_overall_accuracy': 0.9548082239983914, 'test_runtime': 18.3624, 'test_samples_per_second': 47.216, 'test_steps_per_second': 2.995}
```

3.



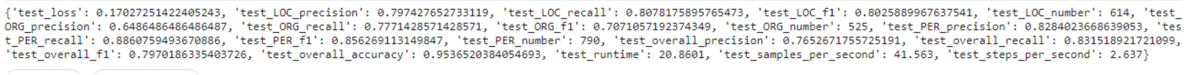
```
args=TrainingArguments(
    output_dir='output_dir',
    per_device_train_batch_size=8,
    per_device_eval_batch_size=8,
    num_train_epochs=3,
    evaluation_strategy = "epoch",
    learning_rate=5e-5,
    weight_decay=0.02
)
```

Output on Validation Data set:



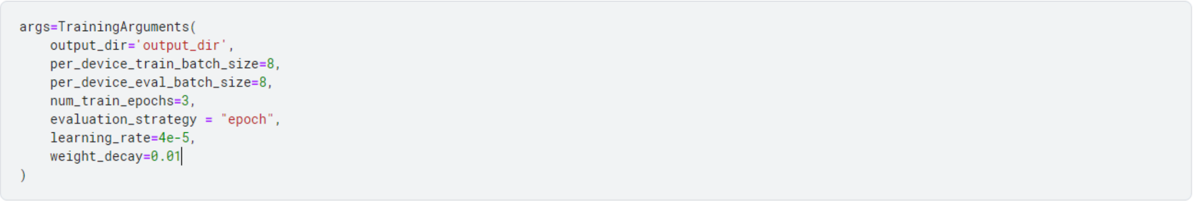
Epoch	Training Loss	Validation Loss	Loc Precision	Loc Recall	Loc F1	Loc Number	Org Precision	Org Recall	Org F1	Org Number	Per Precision	Per Recall	Per F1	Per Number	Overall Precision	Overall Recall	Overall F1	Overall Accuracy
1	0.153100	0.170998	0.810911	0.855772	0.832738	10213	0.681762	0.676885	0.679315	9786	0.805359	0.839042	0.821856	10568	0.769140	0.792718	0.780751	0.947085
2	0.103200	0.182718	0.817899	0.849212	0.833261	10213	0.672278	0.697834	0.684817	9786	0.805091	0.829012	0.816876	10568	0.766749	0.793765	0.780023	0.946727
3	0.074100	0.207823	0.810879	0.850974	0.830443	10213	0.670534	0.685060	0.677719	9786	0.799155	0.823429	0.811111	10568	0.762443	0.788334	0.775172	0.945797

Output on Test Data set:



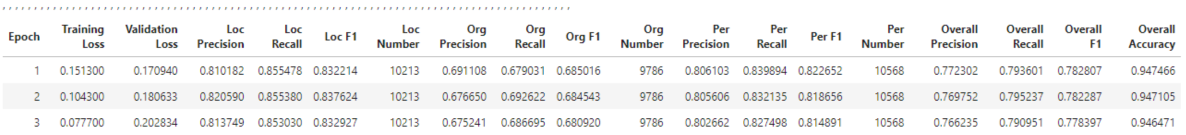
```
{'test_loss': 0.17027251422405243, 'test_LOC_precision': 0.797427652733119, 'test_LOC_recall': 0.8078175895765473, 'test_LOC_f1': 0.8025889967637541, 'test_LOC_number': 614, 'test_
ORG_precision': 0.6486486486486487, 'test_ORG_recall': 0.7771428571428571, 'test_ORG_f1': 0.7071057192374349, 'test_ORG_number': 525, 'test_PER_precision': 0.8284023668639053, 'tes
t_PER_recall': 0.8860759493670886, 'test_PER_f1': 0.856269113149847, 'test_PER_number': 790, 'test_overall_precision': 0.7652671755725191, 'test_overall_recall': 0.831518921721099,
'test_overall_f1': 0.7970186335403726, 'test_overall_accuracy': 0.9536520384054693, 'test_runtime': 20.8601, 'test_samples_per_second': 41.563, 'test_steps_per_second': 2.637}
```

4.



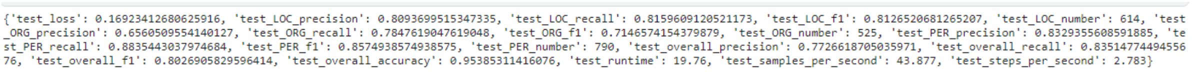
```
args=TrainingArguments(
    output_dir='output_dir',
    per_device_train_batch_size=8,
    per_device_eval_batch_size=8,
    num_train_epochs=3,
    evaluation_strategy = "epoch",
    learning_rate=4e-5,
    weight_decay=0.01
)
```

Output on Validation Data set:



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	0.151300	0.170940	0.810182	0.855478	0.832214	10213	0.691108	0.679031	0.685016	9786	0.806103	0.839894	0.822652	10568	0.772302	0.793601	0.782807	0.947466
2	0.104300	0.180633	0.820590	0.855380	0.837624	10213	0.676650	0.692622	0.684543	9786	0.805606	0.832135	0.818656	10568	0.769752	0.795237	0.782287	0.947105
3	0.077700	0.202834	0.813749	0.853030	0.832927	10213	0.675241	0.686695	0.680920	9786	0.802662	0.827498	0.814891	10568	0.766235	0.790951	0.778397	0.946471

Output on Test Data set:



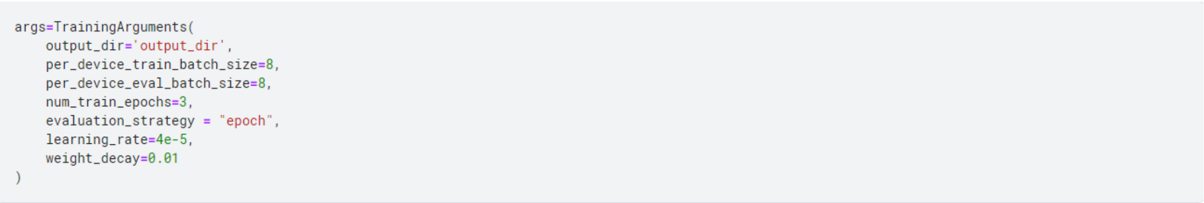
```
{'test_loss': 0.16923412680625916, 'test_LOC_precision': 0.8093699515347335, 'test_LOC_recall': 0.8159609120521173, 'test_LOC_f1': 0.8126520681265207, 'test_LOC_number': 614, 'test
_ORG_precision': 0.6560509554140127, 'test_ORG_recall': 0.7847619047619048, 'test_ORG_f1': 0.7146574154379879, 'test_ORG_number': 525, 'test_PER_precision': 0.8329355608591885, 'te
st_PER_recall': 0.8835443037974684, 'test_PER_f1': 0.8574938574938575, 'test_PER_number': 790, 'test_overall_precision': 0.7726618705035971, 'test_overall_recall': 0.83514774494556
76, 'test_overall_f1': 0.8026905829596414, 'test_overall_accuracy': 0.95385311416076, 'test_runtime': 19.76, 'test_samples_per_second': 43.877, 'test_steps_per_second': 2.783}
```

Conclusion:We can observe that batch size =12 gives better f1 score on validation data set but batch size=8 gives better f1 score on test data set.so we have choosed batch size 8 and then we add weight decay =0.2 but it has decreased the performance so we have reduced that to 0.1 and reduce learning rate then we get overall better performance in 4th iteration.

Output Of IndicBERT Model:

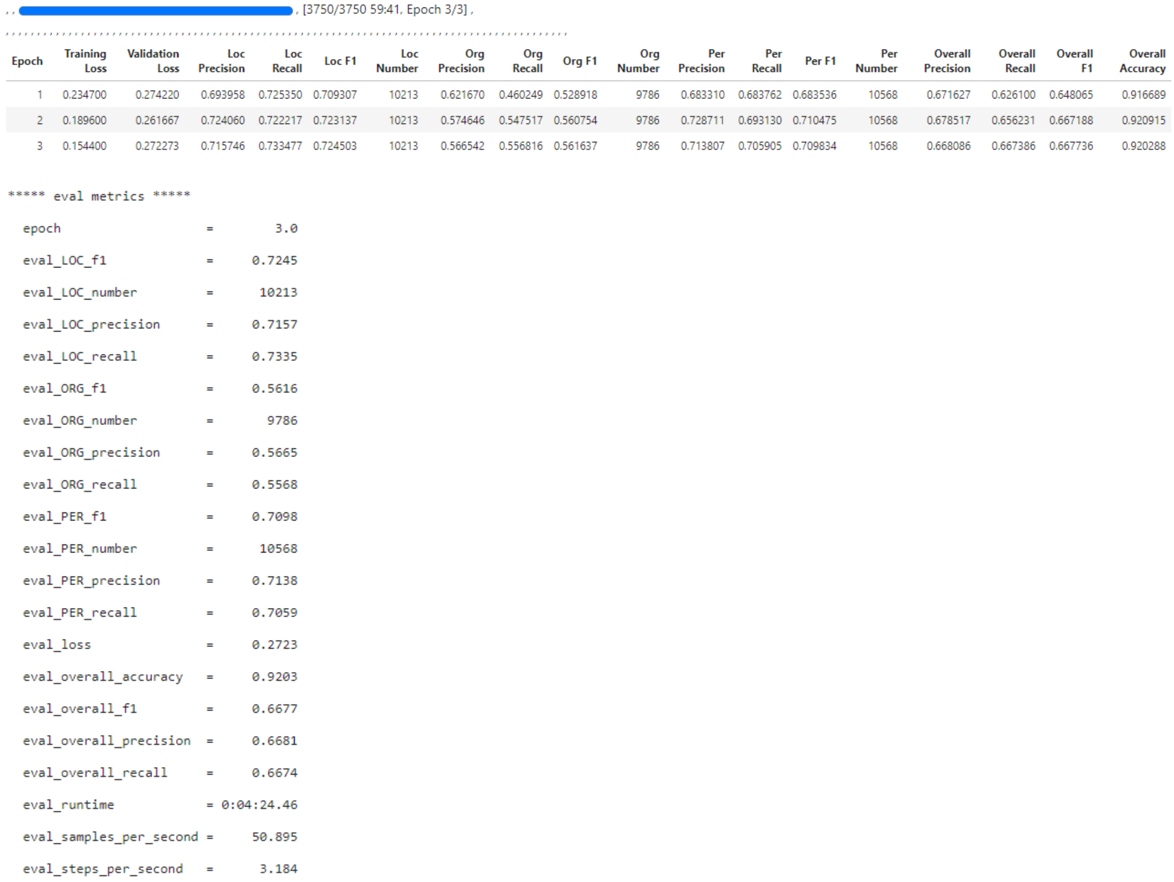
With best finetuned Version

Arguments are:



```
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=4e-5,
    weight_decay=0.01
)
```

Class wise and overall Precision ,Recall and macro f1 score/overall f1 score of Validation Data set

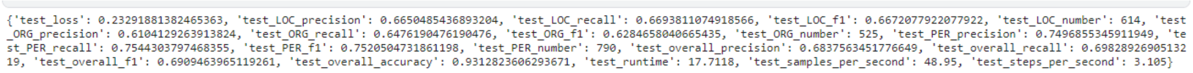


[3750/3750 59:41, Epoch 3/3]

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	0.234700	0.274220	0.693958	0.725350	0.709307	10213	0.621670	0.460249	0.528918	9786	0.683310	0.683762	0.683536	10568	0.671627	0.626100	0.648065	0.916689
2	0.189600	0.261667	0.724060	0.722217	0.723137	10213	0.574646	0.547517	0.560754	9786	0.728711	0.693130	0.710475	10568	0.678517	0.656231	0.667188	0.920915
3	0.154400	0.272273	0.715746	0.733477	0.724503	10213	0.566542	0.556816	0.561637	9786	0.713807	0.705905	0.709834	10568	0.668086	0.667386	0.667736	0.920288

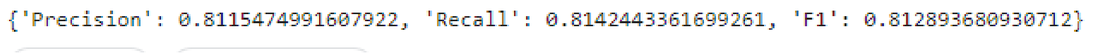
```
***** eval metrics *****
  epoch                   =        3.0
  eval_LOC_f1             =     0.7245
  eval_LOC_number         =      10213
  eval_LOC_precision      =     0.7157
  eval_LOC_recall         =     0.7335
  eval_ORG_f1             =     0.5616
  eval_ORG_number         =       9786
  eval_ORG_precision      =     0.5665
  eval_ORG_recall         =     0.5568
  eval_PER_f1             =     0.7098
  eval_PER_number         =      10568
  eval_PER_precision      =     0.7138
  eval_PER_recall         =     0.7059
  eval_loss               =     0.2723
  eval_overall_accuracy   =     0.9203
  eval_overall_f1         =     0.6677
  eval_overall_precision  =     0.6681
  eval_overall_recall     =     0.6674
  eval_runtime            = 0:04:24.46
  eval_samples_per_second =     50.895
  eval_steps_per_second   =      3.184
```

Class wise and overall Precision, Recall and macro f1 score/overall f1 score of Test Data set



```
{'test_loss': 0.23291881382465363, 'test_LOC_precision': 0.6650485436893204, 'test_LOC_recall': 0.6693811074918566, 'test_LOC_f1': 0.6672077922077922, 'test_LOC_number': 614, 'test
_ORG_precision': 0.6104129263913824, 'test_ORG_recall': 0.6476190476190476, 'test_ORG_f1': 0.6284658040665435, 'test_ORG_number': 525, 'test_PER_precision': 0.7496855345911949, 'te
st_PER_recall': 0.7544303797468355, 'test_PER_f1': 0.7520504731861198, 'test_PER_number': 790, 'test_overall_precision': 0.6837563451776649, 'test_overall_recall': 0.69828926905132
19, 'test_overall_f1': 0.6909463965119261, 'test_overall_accuracy': 0.9312823606293671, 'test_runtime': 17.7118, 'test_samples_per_second': 48.95, 'test_steps_per_second': 3.105}
```

Overall Precision, Recall and macro f1 score/overall f1 score of Training(20000 sentences) Data set

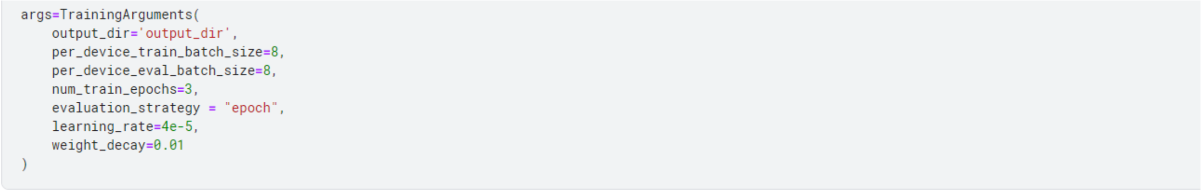


```
{'Precision': 0.8115474991607922, 'Recall': 0.8142443361699261, 'F1': 0.812893680930712}
```

Output Of IndicNER Model:

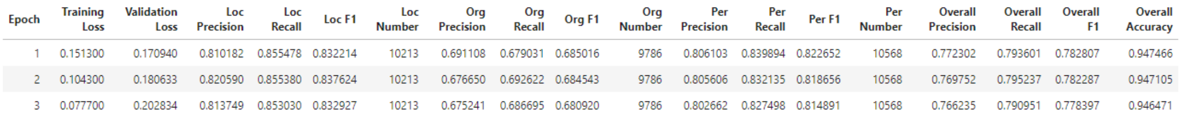
With best finetuned Version

Arguments are:

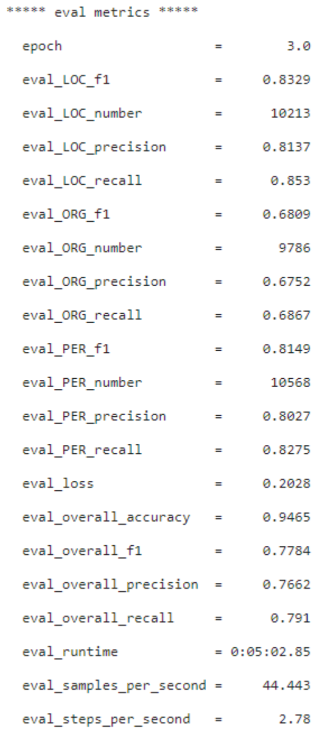


```
args=TrainingArguments(
    output_dir='output_dir',
    per_device_train_batch_size=8,
    per_device_eval_batch_size=8,
    num_train_epochs=3,
    evaluation_strategy = "epoch",
    learning_rate=4e-5,
    weight_decay=0.01
)
```

Class wise and overall Precision ,Recall and macro f1 score/overall f1 score of Validation Data set

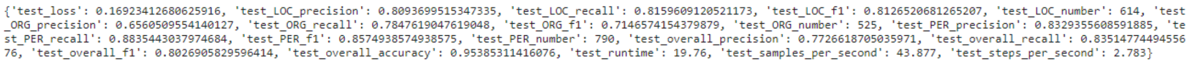


Epoch	Training Loss	Validation Loss	Loc Precision	Loc Recall	Loc F1	Loc Number	Org Precision	Org Recall	Org F1	Org Number	Per Precision	Per Recall	Per F1	Per Number	Overall Precision	Overall Recall	Overall F1	Overall Accuracy
1	0.151300	0.170940	0.810182	0.855478	0.832214	10213	0.691108	0.679031	0.685016	9786	0.806103	0.839894	0.822652	10568	0.772302	0.793601	0.782807	0.947466
2	0.104300	0.180633	0.820590	0.855380	0.837624	10213	0.676650	0.692622	0.684543	9786	0.805606	0.832135	0.818656	10568	0.769752	0.795237	0.782287	0.947105
3	0.077700	0.202834	0.813749	0.853030	0.832927	10213	0.675241	0.686695	0.680920	9786	0.802662	0.827498	0.814891	10568	0.766235	0.790951	0.778397	0.946471



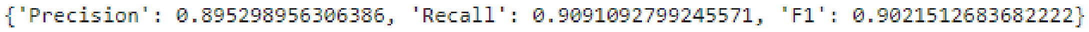
```
***** eval metrics *****

  epoch                   =        3.0

  eval_LOC_f1             =     0.8329

  eval_LOC_number         =      10213

  eval_LOC_precision      =     0.8137

  eval_LOC_recall         =      0.853

  eval_ORG_f1             =     0.6809

  eval_ORG_number         =       9786

  eval_ORG_precision      =     0.6752

  eval_ORG_recall         =     0.6867

  eval_PER_f1             =     0.8149

  eval_PER_number         =      10568

  eval_PER_precision      =     0.8027

  eval_PER_recall         =     0.8275

  eval_loss               =     0.2028

  eval_overall_accuracy   =     0.9465

  eval_overall_f1         =     0.7784

  eval_overall_precision  =     0.7662

  eval_overall_recall     =      0.791

  eval_runtime            = 0:05:02.85

  eval_samples_per_second =     44.443

  eval_steps_per_second   =       2.78
```

Class wise and overall Precision, Recall and macro f1 score/overall f1 score of Test Data set



```
{'test_loss': 0.16923412680625916, 'test_LOC_precision': 0.8093699515347335, 'test_LOC_recall': 0.815960912052l173, 'test_LOC_f1': 0.8126520681265207, 'test_LOC_number': 614, 'test
_ORG_precision': 0.6560509554140127, 'test_ORG_recall': 0.7847619047619048, 'test_ORG_f1': 0.7146574154379879, 'test_ORG_number': 525, 'test_PER_precision': 0.8329355608591885, 'te
st_PER_recall': 0.8835443037974684, 'test_PER_f1': 0.8574938574938575, 'test_PER_number': 790, 'test_overall_precision': 0.7726618705035971, 'test_overall_recall': 0.83514774494556
76, 'test_overall_f1': 0.8026905829596414, 'test_overall_accuracy': 0.95385311416076, 'test_runtime': 19.76, 'test_samples_per_second': 43.877, 'test_steps_per_second': 2.783}
```

Overall Precision, Recall and macro f1 score/overall f1 score of Training(20000 sentences) Data set



```
{'Precision': 0.895298956306386, 'Recall': 0.9091092799245571, 'F1': 0.9021512683682222}
```