CHATGPT

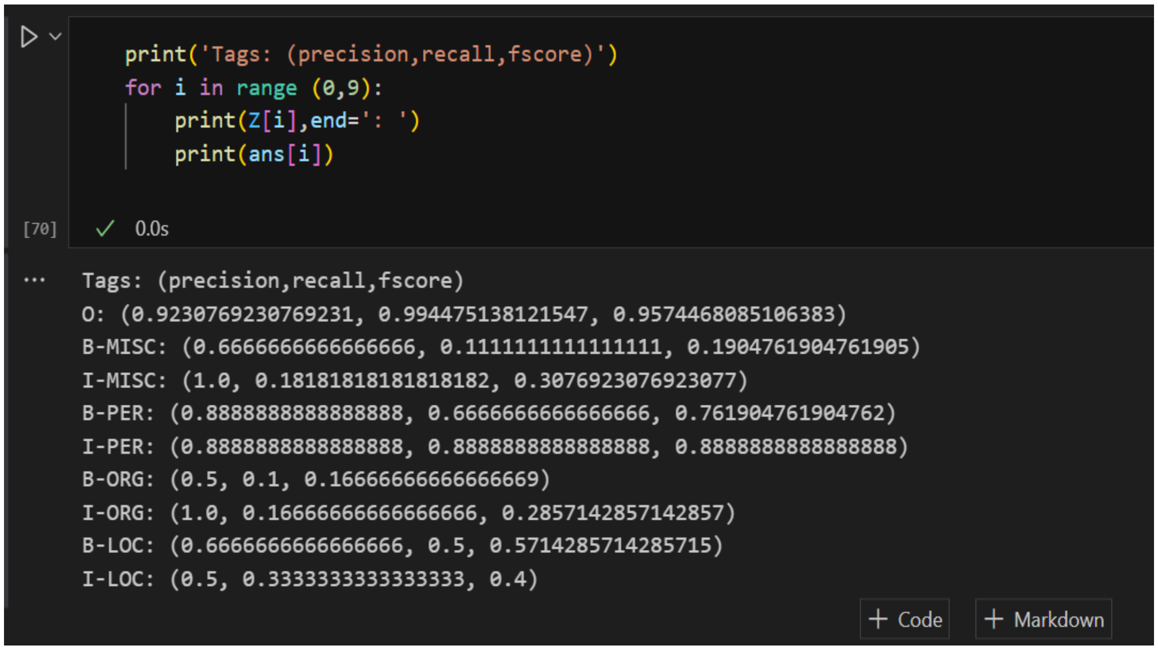
COMPARISON OF CHAT GPT VS MANUAL ANSWERS

This report compares the Named Entity Recognition (NER) tagging done by me and the responses generated by ChatGPT for a set of 25 sentences. Here's a summary of the findings:

- There were discrepancies in entity recognition between my tagging and ChatGPT's answers.
- I identified specific entities (e.g., names of individuals and locations) that ChatGPT missed.
- ChatGPT's responses missed tagging entities in several sentences, indicating areas for improvement.
- ChatGPT provided direct answers without recognizing the context of the sentence, resulting in different tagging for the same words based on the sentence's context.

In conclusion, while both approaches have their strengths and weaknesses, a combined effort of human review and AI assistance could enhance the accuracy of NER tagging in natural language processing tasks.

PRECISION RECALL AND F SCORE OF COMPARISON BETWEEN MANUAL AND CHATGPT ANSWERS

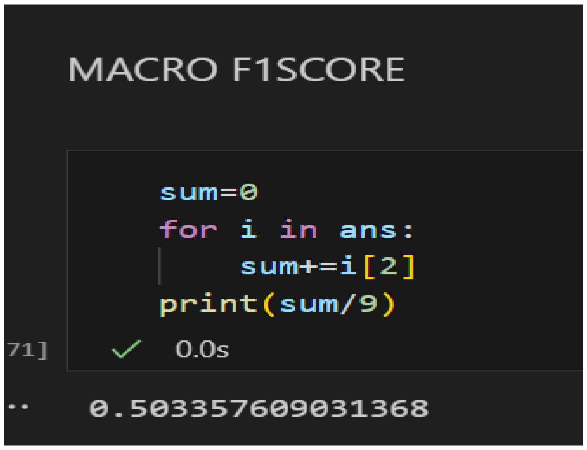


```
print('Tags: (precision,recall,fscore)')
for i in range (0,9):
    print(Z[i],end=': ')
    print(ans[i])
```

[70] ✓ 0.0s

```
Tags: (precision,recall,fscore)
O: (0.9230769230769231, 0.994475138121547, 0.9574468085106383)
B-MISC: (0.6666666666666666, 0.1111111111111111, 0.1904761904761905)
I-MISC: (1.0, 0.18181818181818182, 0.3076923076923077)
B-PER: (0.8888888888888888, 0.6666666666666666, 0.761904761904762)
I-PER: (0.8888888888888888, 0.8888888888888888, 0.8888888888888888)
B-ORG: (0.5, 0.1, 0.16666666666666669)
I-ORG: (1.0, 0.16666666666666666, 0.2857142857142857)
B-LOC: (0.6666666666666666, 0.5, 0.5714285714285715)
I-LOC: (0.5, 0.3333333333333333, 0.4)
```

MACRO F1 SCORE OF COMPARISON BETWEEN MANUAL AND CHATGPT ANSWERS



MACRO F1SCORE

```
sum=0
for i in ans:
    sum+=i[2]
print(sum/9)
```

```
71]  ✓ 0.0s
```

```
0.503357609031368
```

INDICBERT

FINE TUINING OF INDICBERT MODEL

PARAMETERS

per_device_train_batch_size: This parameter determines the number of training samples per batch for each device (e.g., GPU or CPU). Setting it to 8 means that each device will process 8 training samples in parallel during training.I have set to 8 and 16

per_device_eval_batch_size: Similar to per_device_train_batch_size, this parameter determines the number of evaluation samples per batch for each device during evaluation (e.g., validation or testing). Setting it to 8 means that each device will process 8 evaluation samples in parallel during evaluation.
I have set to 8 and 16

num_train_epochs: This parameter specifies the number of epochs (complete passes through the training dataset) during training. Setting it to 3 means that the model will go through the entire training dataset three times during the fine-tuning process.I have set to 3

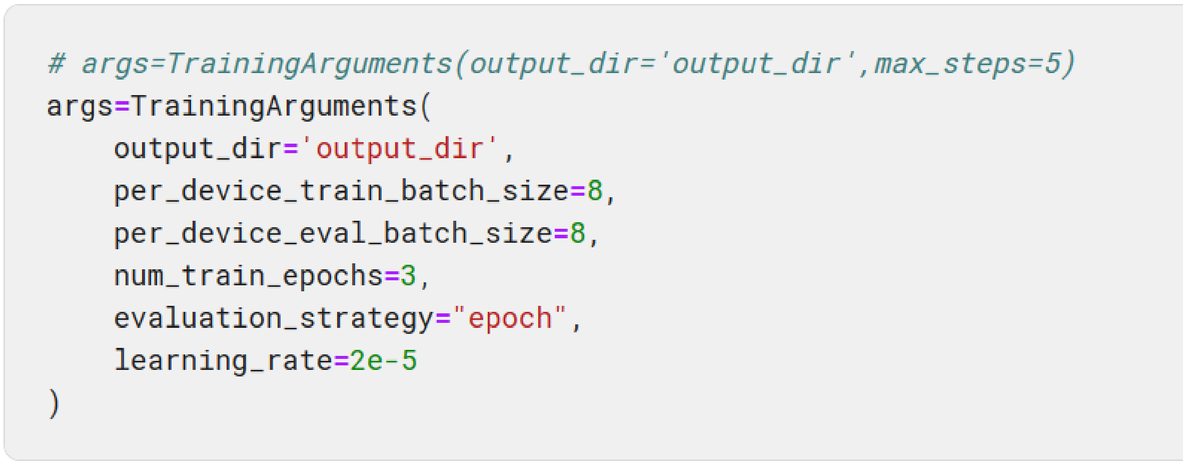
evaluation_strategy: This parameter determines when to perform evaluation during training. Setting it to "epoch" means that evaluation will be performed after each training epoch.

learning_rate: This parameter controls the step size at which the model weights are updated during training. Setting it to 4e-5 means that the model's weights will be updated with a learning rate of 4x10^-5.I have set this to 4x10^-5 and 2x10^-5.

weight_decay: This parameter adds a penalty term to the model's loss function to prevent overfitting by discouraging large weights. Setting it to 0.01 means that a weight decay factor of 0.01 will be applied during training.

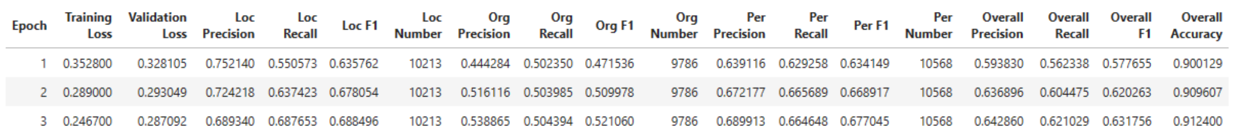
1

PARAMETERS SET:



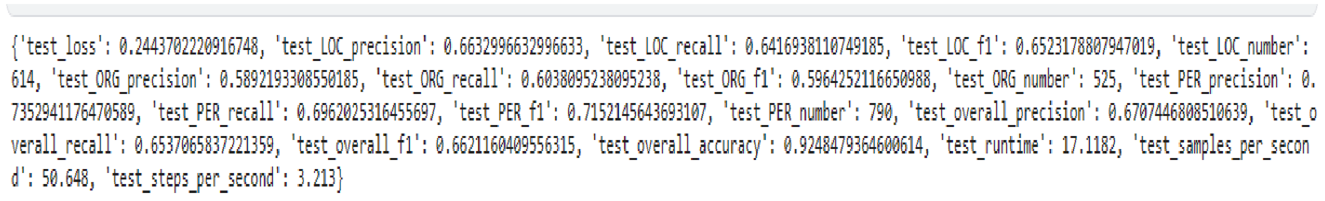
```
# args=TrainingArguments(output_dir='output_dir',max_steps=5)
args=TrainingArguments(
    output_dir='output_dir',
    per_device_train_batch_size=8,
    per_device_eval_batch_size=8,
    num_train_epochs=3,
    evaluation_strategy="epoch",
    learning_rate=2e-5
)
```

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	0.352800	0.328105	0.752140	0.550573	0.635762	10213	0.444284	0.502350	0.471536	9786	0.639116	0.629258	0.634149	10568	0.593830	0.562338	0.577655	0.900129
2	0.289000	0.293049	0.724218	0.637423	0.678054	10213	0.516116	0.503985	0.509978	9786	0.672177	0.665689	0.668917	10568	0.636896	0.604475	0.620263	0.909607
3	0.246700	0.287092	0.689340	0.687653	0.688496	10213	0.538865	0.504394	0.521060	9786	0.689913	0.664648	0.677045	10568	0.642860	0.621029	0.631756	0.912400

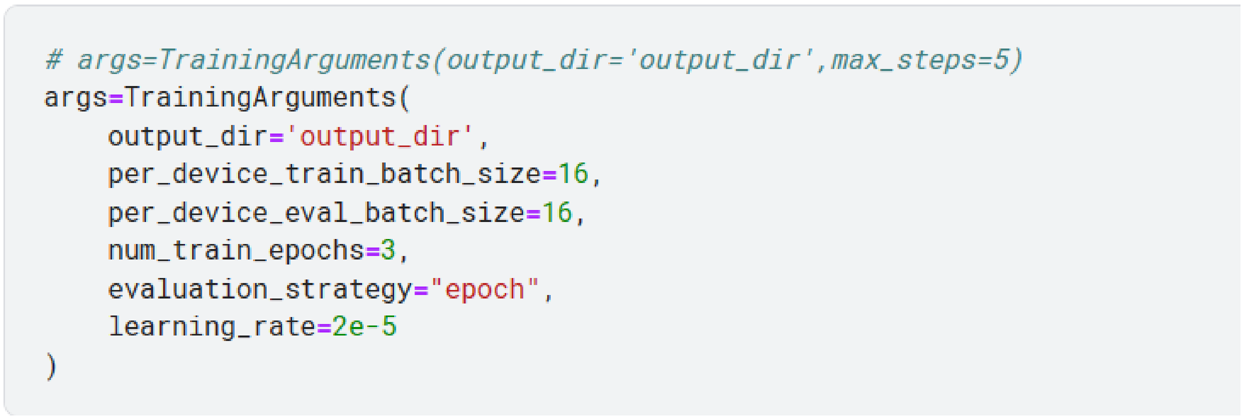
OUTPUT ON TEST DATA SET:



```
{'test_loss': 0.2443702220916748, 'test_LOC_precision': 0.6632996632996633, 'test_LOC_recall': 0.6416938110749185, 'test_LOC_f1': 0.6523178807947019, 'test_LOC_number':
614, 'test_ORG_precision': 0.5892193308550185, 'test_ORG_recall': 0.6038095238095238, 'test_ORG_f1': 0.5964252116650988, 'test_ORG_number': 525, 'test_PER_precision': 0.
7352941176470589, 'test_PER_recall': 0.6962025316455697, 'test_PER_f1': 0.7152145643693107, 'test_PER_number': 790, 'test_overall_precision': 0.6707446808510639, 'test_o
verall_recall': 0.6537065837221359, 'test_overall_f1': 0.6621160409556315, 'test_overall_accuracy': 0.9248479364600614, 'test_runtime': 17.1182, 'test_samples_per_secon
d': 50.648, 'test_steps_per_second': 3.213}
```

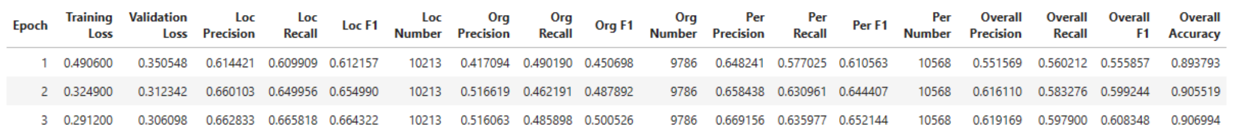
2

PARAMETER SET:



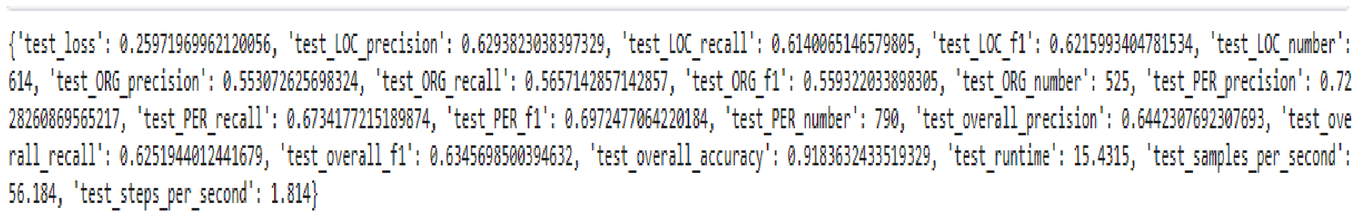
```
# args=TrainingArguments(output_dir='output_dir',max_steps=5)
args=TrainingArguments(
    output_dir='output_dir',
    per_device_train_batch_size=16,
    per_device_eval_batch_size=16,
    num_train_epochs=3,
    evaluation_strategy="epoch",
    learning_rate=2e-5
)
```

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	0.490600	0.350548	0.614421	0.609909	0.612157	10213	0.417094	0.490190	0.450698	9786	0.648241	0.577025	0.610563	10568	0.551569	0.560212	0.555857	0.893793
2	0.324900	0.312342	0.660103	0.649956	0.654990	10213	0.516619	0.462191	0.487892	9786	0.658438	0.630961	0.644407	10568	0.616110	0.583276	0.599244	0.905519
3	0.291200	0.306098	0.662833	0.665818	0.664322	10213	0.516063	0.485898	0.500526	9786	0.669156	0.635977	0.652144	10568	0.619169	0.597900	0.608348	0.906994

OUTPUT ON TEST DATA SET:



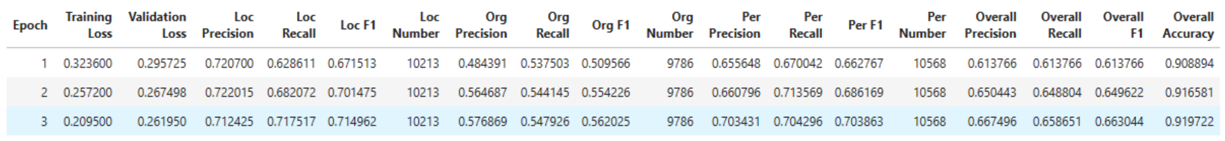
```
{'test_loss': 0.25971969962120056, 'test_LOC_precision': 0.6293823038397329, 'test_LOC_recall': 0.6140065146579805, 'test_LOC_f1': 0.6215993404781534, 'test_LOC_number':
614, 'test_ORG_precision': 0.553072625698324, 'test_ORG_recall': 0.5657142857142857, 'test_ORG_f1': 0.559322033898305, 'test_ORG_number': 525, 'test_PER_precision': 0.72
28260869565217, 'test_PER_recall': 0.6734177215189874, 'test_PER_f1': 0.6972477064220184, 'test_PER_number': 790, 'test_overall_precision': 0.6442307692307693, 'test_ove
rall_recall': 0.6251944012441679, 'test_overall_f1': 0.6345698500394632, 'test_overall_accuracy': 0.9183632433519329, 'test_runtime': 15.4315, 'test_samples_per_second':
56.184, 'test_steps_per_second': 1.814}
```

3

PARAMETER SET:

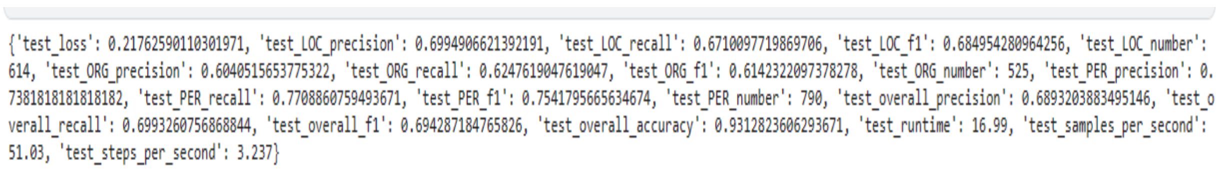
```
# args=TrainingArguments(output_dir='output_dir',max_steps=5)
args=TrainingArguments(
    output_dir='output_dir',
    per_device_train_batch_size=8,
    per_device_eval_batch_size=8,
    num_train_epochs=3,
    evaluation_strategy="epoch",
    learning_rate=4e-5
)
```

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	0.323600	0.295725	0.720700	0.628611	0.671513	10213	0.484391	0.537503	0.509566	9786	0.655648	0.670042	0.662767	10568	0.613766	0.613766	0.613766	0.908894
2	0.257200	0.267498	0.722015	0.682072	0.701475	10213	0.564687	0.544145	0.554226	9786	0.660796	0.713569	0.686169	10568	0.650443	0.648804	0.649622	0.916581
3	0.209500	0.261950	0.712425	0.717517	0.714962	10213	0.576869	0.547926	0.562025	9786	0.703431	0.704296	0.703863	10568	0.667496	0.658651	0.663044	0.919722

OUTPUT ON TEST DATA SET:



```
{'test_loss': 0.21762590110301971, 'test_LOC_precision': 0.6994906621392191, 'test_LOC_recall': 0.6710097719869706, 'test_LOC_f1': 0.684954280964256, 'test_LOC_number':
614, 'test_ORG_precision': 0.6040515653775322, 'test_ORG_recall': 0.6247619047619047, 'test_ORG_f1': 0.6142322097378278, 'test_ORG_number': 525, 'test_PER_precision': 0.
7381818181818182, 'test_PER_recall': 0.7708860759493671, 'test_PER_f1': 0.7541795665634674, 'test_PER_number': 790, 'test_overall_precision': 0.6893203883495146, 'test_o
verall_recall': 0.6993260756868844, 'test_overall_f1': 0.694287184765826, 'test_overall_accuracy': 0.9312823606293671, 'test_runtime': 16.99, 'test_samples_per_second':
51.03, 'test_steps_per_second': 3.237}
```

4

PARAMETER SET:

```python
from transformers import TrainingArguments

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   # Set the weight decay value here
)
```

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	0.323700	0.297312	0.726742	0.621854	0.670220	10213	0.479573	0.527795	0.502530	9786	0.685258	0.658024	0.671365	10568	0.622787	0.604246	0.613377	0.909288
2	0.258300	0.267385	0.718898	0.682366	0.700156	10213	0.550037	0.536378	0.543122	9786	0.677573	0.713285	0.694971	10568	0.650682	0.646318	0.648493	0.917162
3	0.208400	0.263925	0.704824	0.725252	0.714892	10213	0.573421	0.533517	0.552750	9786	0.700608	0.709027	0.704792	10568	0.663862	0.658259	0.661049	0.919292

OUTPUT ON TEST DATA SET:

```
{'test_loss': 0.21802595257759094, 'test_LOC_precision': 0.6930860033726813, 'test_LOC_recall': 0.6693811074918566, 'test_LOC_f1': 0.6810273405136702, 'test_LOC_num
ber': 614, 'test_ORG_precision': 0.6236363636363637, 'test_ORG_recall': 0.6533333333333333, 'test_ORG_f1': 0.638139534883721, 'test_ORG_number': 525, 'test_PER_prec
ision': 0.7237977805178791, 'test_PER_recall': 0.7430379746835443, 'test_PER_f1': 0.7332916926920675, 'test_PER_number': 790, 'test_overall_precision': 0.6862845445
240532, 'test_overall_recall': 0.6951788491446346, 'test_overall_f1': 0.6907030646407417, 'test_overall_accuracy': 0.9324385462222893, 'test_runtime': 16.4676, 'tes
t_samples_per_second': 52.649, 'test_steps_per_second': 3.34}
```

CONCLUSION

Since we got highest accuracy on test and validation data by setting the parameter batch size = 8 and learning rate = 4e-5 and weight decay =0.01 so it works best for us although the differences are very minor.

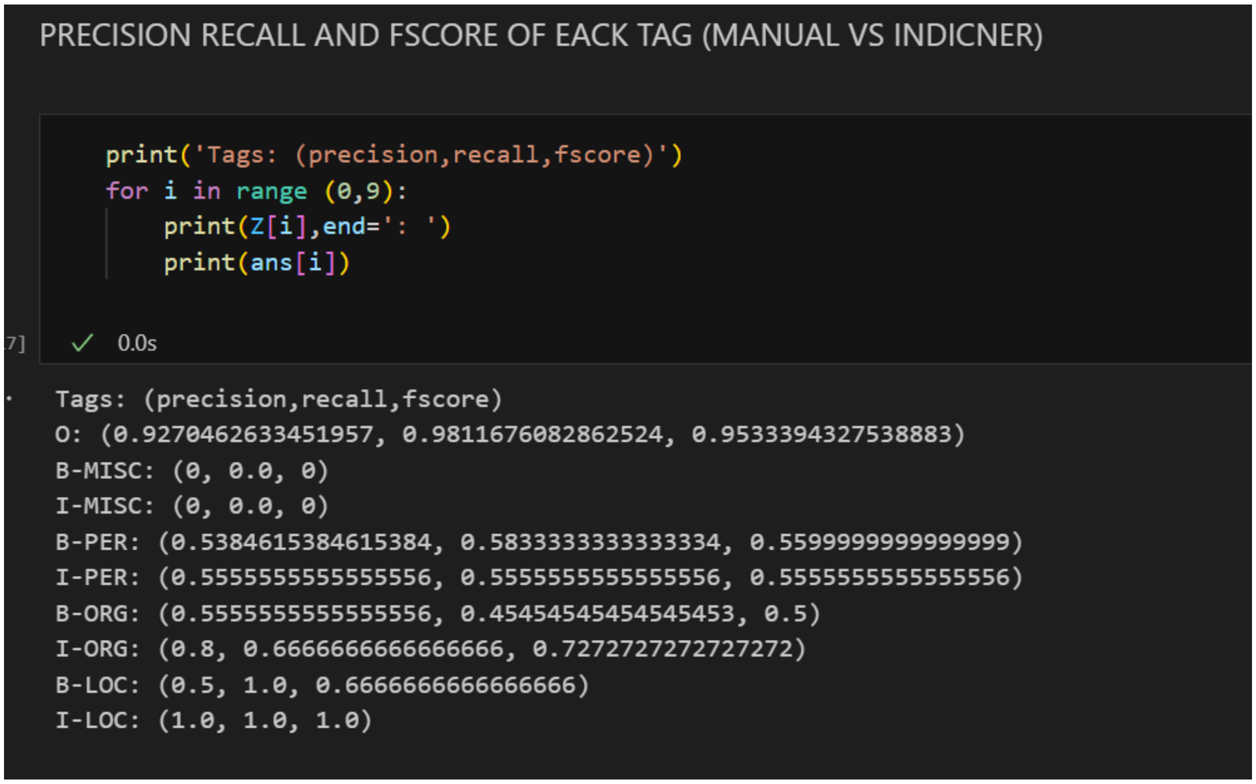
COMPARISON OF INDICBERT VS MANUAL ANSWERS

Based on the output provided by the IndicBERT model for Named Entity Recognition (NER) tagging:

- The model correctly identified entities such as names of individuals (राजीव गाँधी, अनिल कंसल), locations (e.g., दिल्ली), and organizations (जेडीए, नीति आयोग).
- However, there are instances where the model missed tagging certain entities or misclassified them, such as not recognizing '68' as an entity or misclassifying 'गाँधी' as an individual name instead of being part of the organization name 'राजीव गाँधी'.
- Overall, while the model showed good performance in identifying some entities, there is room for improvement, particularly in handling ambiguous cases or rare entities.

In conclusion, the IndicBERT model demonstrates promising capabilities in NER tagging for the given text but may benefit from further fine-tuning and refinement to improve accuracy, especially in handling diverse and complex entity types.

PRECISION RECALL AND F SCORE OF COMPARISON BETWEEN MANUAL AND INDICBERT ANSWERS



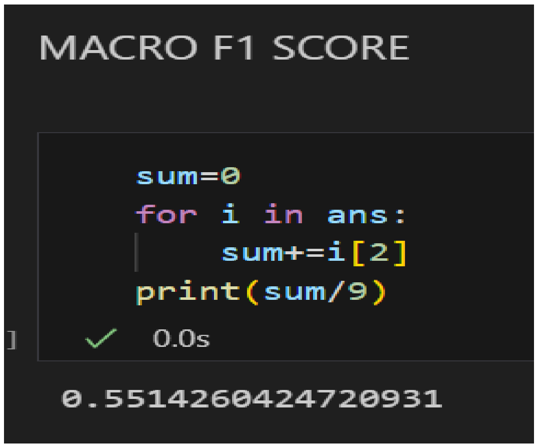
PRECISION RECALL AND FSCORE OF EACK TAG (MANUAL VS INDICNER)

```
print('Tags: (precision,recall,fscore)')
for i in range (0,9):
    print(Z[i],end=': ')
    print(ans[i])
```

✓ 0.0s

```
Tags: (precision,recall,fscore)
O: (0.9270462633451957, 0.9811676082862524, 0.9533394327538883)
B-MISC: (0, 0.0, 0)
I-MISC: (0, 0.0, 0)
B-PER: (0.5384615384615384, 0.5833333333333334, 0.5599999999999999)
I-PER: (0.5555555555555556, 0.5555555555555556, 0.5555555555555556)
B-ORG: (0.5555555555555556, 0.45454545454545453, 0.5)
I-ORG: (0.8, 0.6666666666666666, 0.7272727272727272)
B-LOC: (0.5, 1.0, 0.6666666666666666)
I-LOC: (1.0, 1.0, 1.0)
```

MACRO F1 SCORE OF COMPARISON BETWEEN MANUAL AND CHATGPT ANSWERS


MACRO F1 SCORE

```
sum=0
for i in ans:
    sum+=i[2]
print(sum/9)
```

✓ 0.0s

```
0.5514260424720931
```

INDICNER

FINE TUINING OF INDICNER MODEL

PARAMETERS

per_device_train_batch_size: This parameter determines the number of training samples per batch for each device (e.g., GPU or CPU). Setting it to 8 means that each device will process 8 training samples in parallel during training.I have set to 8 and 16

per_device_eval_batch_size: Similar to per_device_train_batch_size, this parameter determines the number of evaluation samples per batch for each device during evaluation (e.g., validation or testing). Setting it to 8 means that each device will process 8 evaluation samples in parallel during evaluation.
I have set to 8 and 16

num_train_epochs: This parameter specifies the number of epochs (complete passes through the training dataset) during training. Setting it to 3 means that the model will go through the entire training dataset three times during the fine-tuning process.I have set to 3

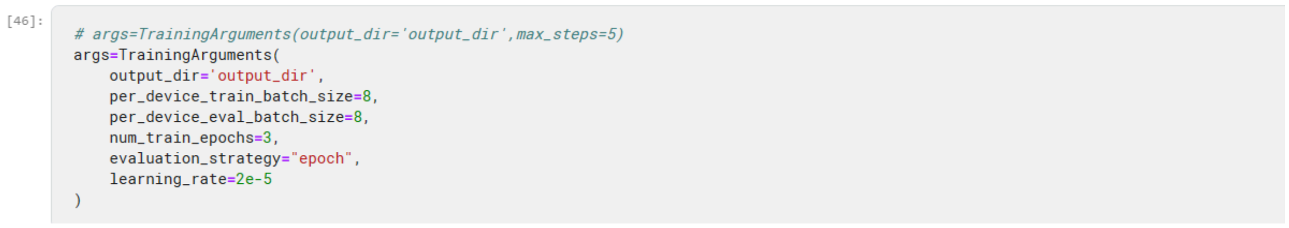
evaluation_strategy: This parameter determines when to perform evaluation during training. Setting it to "epoch" means that evaluation will be performed after each training epoch.

learning_rate: This parameter controls the step size at which the model weights are updated during training. Setting it to 4e-5 means that the model's weights will be updated with a learning rate of 4x10^-5.I have set this to 4x10^-5 and 2x10^-5.

weight_decay: This parameter adds a penalty term to the model's loss function to prevent overfitting by discouraging large weights. Setting it to 0.01 means that a weight decay factor of 0.01 will be applied during training.

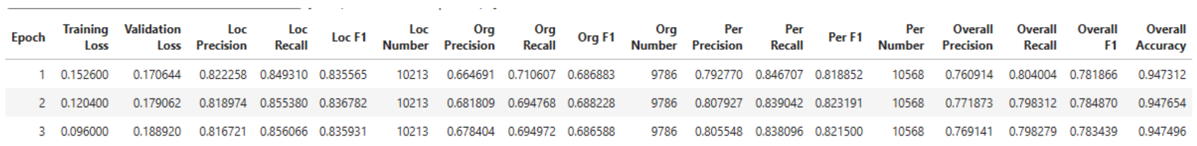
1

PARAMETER SET :



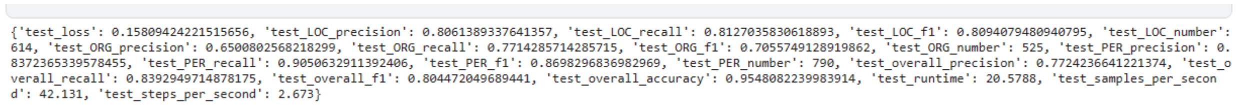
```
[46]:
# args=TrainingArguments(output_dir='output_dir',max_steps=5)
args=TrainingArguments(
    output_dir='output_dir',
    per_device_train_batch_size=8,
    per_device_eval_batch_size=8,
    num_train_epochs=3,
    evaluation_strategy="epoch",
    learning_rate=2e-5
)
```

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	0.152600	0.170644	0.822258	0.849310	0.835565	10213	0.664691	0.710607	0.686883	9786	0.792770	0.846707	0.818852	10568	0.760914	0.804004	0.781866	0.947312
2	0.120400	0.179062	0.818974	0.855380	0.836782	10213	0.681809	0.694768	0.688228	9786	0.807927	0.839042	0.823191	10568	0.771873	0.798312	0.784870	0.947654
3	0.096000	0.188920	0.816721	0.856066	0.835931	10213	0.678404	0.694972	0.686588	9786	0.805548	0.838096	0.821500	10568	0.769141	0.798279	0.783439	0.947496

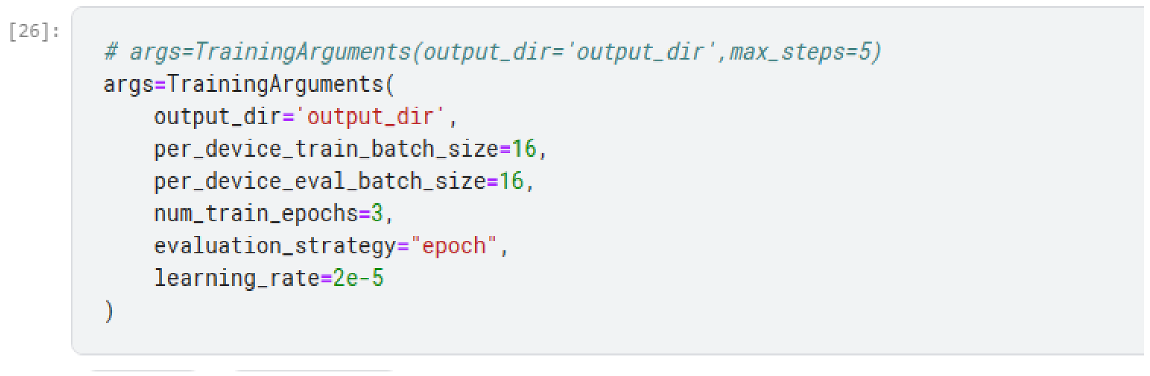
OUTPUT ON TEST DATA SET:



```
{'test_loss': 0.15809424221515656, 'test_LOC_precision': 0.8061389337641357, 'test_LOC_recall': 0.8127035830618893, 'test_LOC_f1': 0.8094079480940795, 'test_LOC_number':
614, 'test_ORG_precision': 0.6500802568218299, 'test_ORG_recall': 0.7714285714285715, 'test_ORG_f1': 0.7055749128919862, 'test_ORG_number': 525, 'test_PER_precision': 0.
8372365339578455, 'test_PER_recall': 0.9050632911392406, 'test_PER_f1': 0.8698296836982969, 'test_PER_number': 790, 'test_overall_precision': 0.7724236641221374, 'test_o
verall_recall': 0.8392949714878175, 'test_overall_f1': 0.804472049689441, 'test_overall_accuracy': 0.9548082239983914, 'test_runtime': 20.5788, 'test_samples_per_secon
d': 42.131, 'test_steps_per_second': 2.673}
```

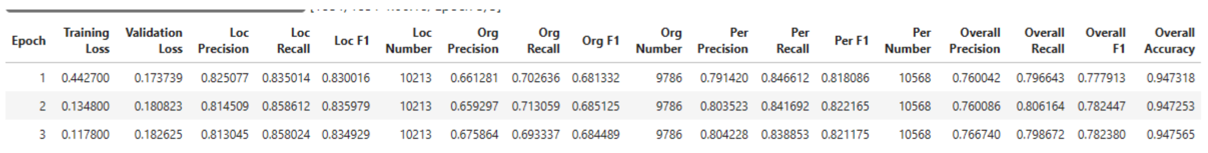
2

PARAMETER SET :



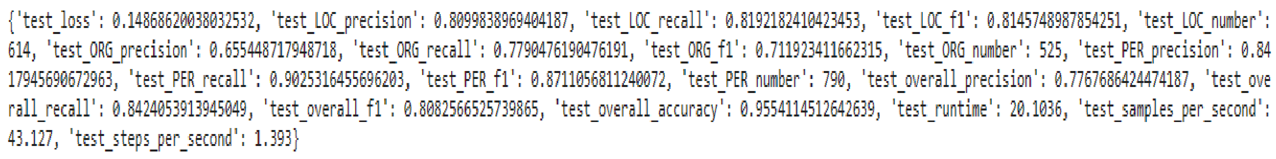
```
[26]:
# args=TrainingArguments(output_dir='output_dir',max_steps=5)
args=TrainingArguments(
    output_dir='output_dir',
    per_device_train_batch_size=16,
    per_device_eval_batch_size=16,
    num_train_epochs=3,
    evaluation_strategy="epoch",
    learning_rate=2e-5
)
```

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	0.442700	0.173739	0.825077	0.835014	0.830016	10213	0.661281	0.702636	0.681332	9786	0.791420	0.846612	0.818086	10568	0.760042	0.796643	0.777913	0.947318
2	0.134800	0.180823	0.814509	0.858612	0.835979	10213	0.659297	0.713059	0.685125	9786	0.803523	0.841692	0.822165	10568	0.760086	0.806164	0.782447	0.947253
3	0.117800	0.182625	0.813045	0.858024	0.834929	10213	0.675864	0.693337	0.684489	9786	0.804228	0.838853	0.821175	10568	0.766740	0.798672	0.782380	0.947565

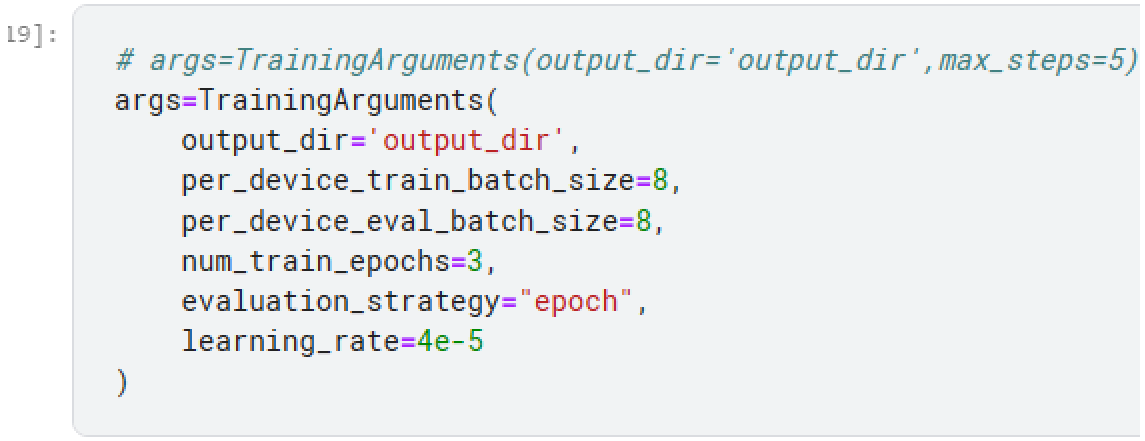
OUTPUT ON TEST DATA SET:



```
{'test_loss': 0.14868620038032532, 'test_LOC_precision': 0.8099838969404187, 'test_LOC_recall': 0.8192182410423453, 'test_LOC_f1': 0.8145748987854251, 'test_LOC_number':
614, 'test_ORG_precision': 0.655448717948718, 'test_ORG_recall': 0.7790476190476191, 'test_ORG_f1': 0.711923411662315, 'test_ORG_number': 525, 'test_PER_precision': 0.84
17945690672963, 'test_PER_recall': 0.9025316455696203, 'test_PER_f1': 0.8711056811240072, 'test_PER_number': 790, 'test_overall_precision': 0.7767686424474187, 'test_ove
rall_recall': 0.8424053913945049, 'test_overall_f1': 0.8082566525739865, 'test_overall_accuracy': 0.9554114512642639, 'test_runtime': 20.1036, 'test_samples_per_second':
43.127, 'test_steps_per_second': 1.393}
```

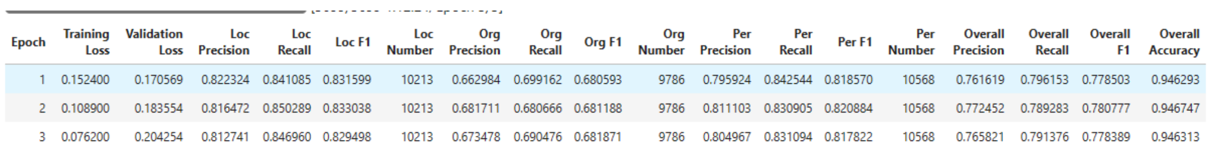
3

PARAMETER SET :



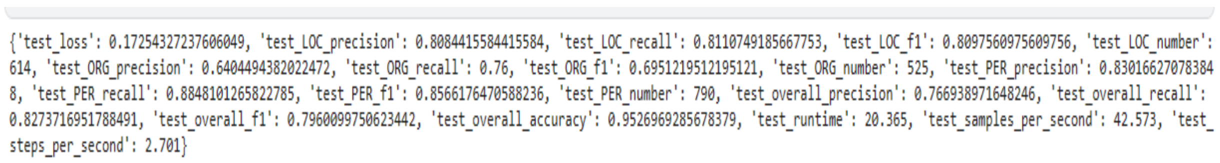
```
19]:
# args=TrainingArguments(output_dir='output_dir',max_steps=5)
args=TrainingArguments(
    output_dir='output_dir',
    per_device_train_batch_size=8,
    per_device_eval_batch_size=8,
    num_train_epochs=3,
    evaluation_strategy="epoch",
    learning_rate=4e-5
)
```

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	0.152400	0.170569	0.822324	0.841085	0.831599	10213	0.662984	0.699162	0.680593	9786	0.795924	0.842544	0.818570	10568	0.761619	0.796153	0.778503	0.946293
2	0.108900	0.183554	0.816472	0.850289	0.833038	10213	0.681711	0.680666	0.681188	9786	0.811103	0.830905	0.820884	10568	0.772452	0.789283	0.780777	0.946747
3	0.076200	0.204254	0.812741	0.846960	0.829498	10213	0.673478	0.690476	0.681871	9786	0.804967	0.831094	0.817822	10568	0.765821	0.791376	0.778389	0.946313

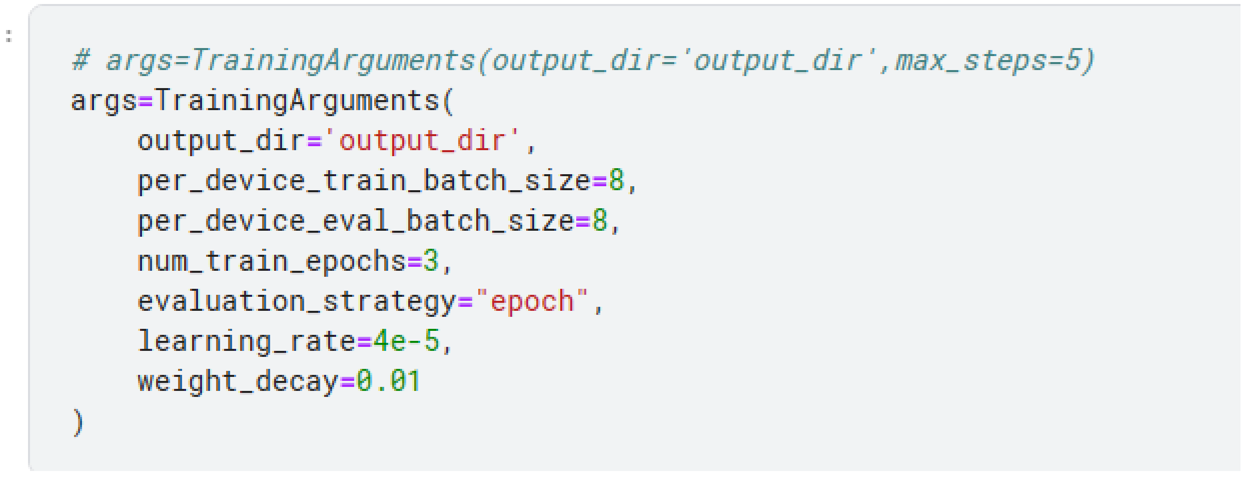
OUTPUT ON TEST DATA SET:



```
{'test_loss': 0.17254327237606049, 'test_LOC_precision': 0.8084415584415584, 'test_LOC_recall': 0.8110749185667753, 'test_LOC_f1': 0.8097560975609756, 'test_LOC_number':
614, 'test_ORG_precision': 0.6404494382022472, 'test_ORG_recall': 0.76, 'test_ORG_f1': 0.6951219512195121, 'test_ORG_number': 525, 'test_PER_precision': 0.830166270783840
8, 'test_PER_recall': 0.8848101265822785, 'test_PER_f1': 0.8566176470588236, 'test_PER_number': 790, 'test_overall_precision': 0.766938971648246, 'test_overall_recall':
0.827371695178849l, 'test_overall_f1': 0.7960099750623442, 'test_overall_accuracy': 0.9526969285678379, 'test_runtime': 20.365, 'test_samples_per_second': 42.573, 'test_
steps_per_second': 2.701}
```

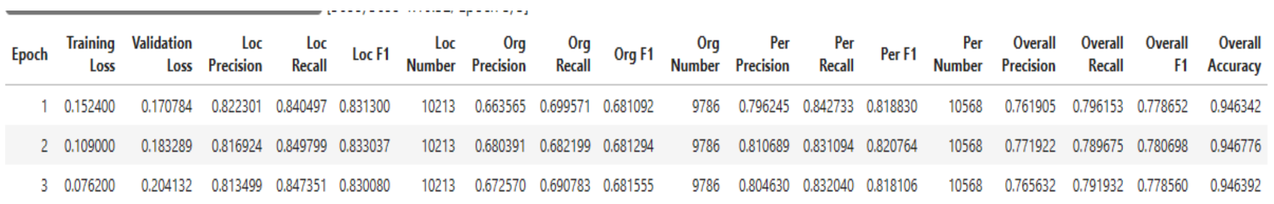
4

PARAMETER SET :



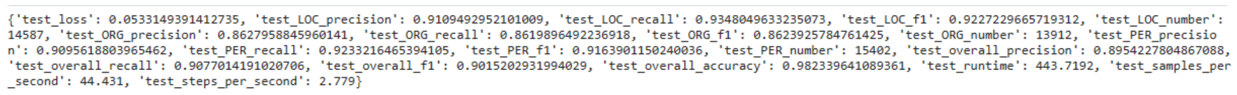
```
# args=TrainingArguments(output_dir='output_dir',max_steps=5)
args=TrainingArguments(
    output_dir='output_dir',
    per_device_train_batch_size=8,
    per_device_eval_batch_size=8,
    num_train_epochs=3,
    evaluation_strategy="epoch",
    learning_rate=4e-5,
    weight_decay=0.01
)
```

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	0.152400	0.170784	0.822301	0.840497	0.831300	10213	0.663565	0.699571	0.681092	9786	0.796245	0.842733	0.818830	10568	0.761905	0.796153	0.778652	0.946342
2	0.109000	0.183289	0.816924	0.849799	0.833037	10213	0.680391	0.682199	0.681294	9786	0.810689	0.831094	0.820764	10568	0.771922	0.789675	0.780698	0.946776
3	0.076200	0.204132	0.813499	0.847351	0.830080	10213	0.672570	0.690783	0.681555	9786	0.804630	0.832040	0.818106	10568	0.765632	0.791932	0.778560	0.946392

OUTPUT ON TEST DATA SET:



```
{'test_loss': 0.0533149391412735, 'test_LOC_precision': 0.9109492952101009, 'test_LOC_recall': 0.9348049633235073, 'test_LOC_f1': 0.9227229665719312, 'test_LOC_number':
14587, 'test_ORG_precision': 0.8627958845960141, 'test_ORG_recall': 0.8619896492236918, 'test_ORG_f1': 0.8623925784761425, 'test_ORG_number': 13912, 'test_PER_precisio
n': 0.9095618803965462, 'test_PER_recall': 0.9233216465394105, 'test_PER_f1': 0.9163901150240036, 'test_PER_number': 15402, 'test_overall_precision': 0.8954227804867088,
'test_overall_recall': 0.9077014191020706, 'test_overall_f1': 0.9015202931994029, 'test_overall_accuracy': 0.982339641089361, 'test_runtime': 443.7192, 'test_samples_per
_second': 44.431, 'test_steps_per_second': 2.779}
```

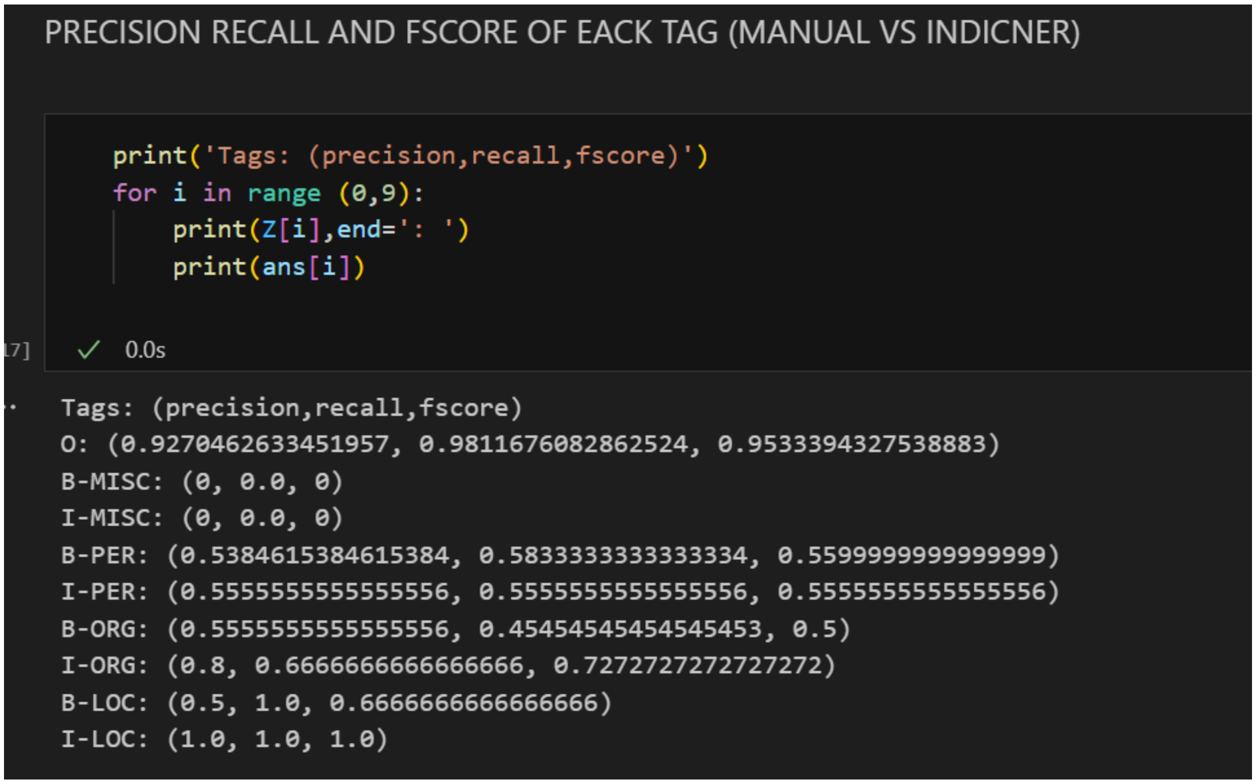
CONCLUSION

By varying the different parameters such as batch size and learning rate and weight decay the overall accuracy is not affecting much although optimal values chosen by me are batch size=8, Learning rate=4e-5,and weight decay =0.01.

COMPARISON OF INDICNER VS MANUAL ANSWERS

The IndicNER model output shows correct identification and tagging of various named entities in the given text. It correctly identifies names of individuals, such as "राजे" and "छह" in the first sentence, "राजीव गाँधी" in the third sentence, and "विवेकानंद रेड्डी" in the sixth sentence. It also correctly identifies locations like "दिल्ली" in the second sentence.
However, there are instances where the model misclassified entities or misses tagging certain entities. For example, it incorrectly tags "68" as a person name in the sixth sentence and fails to recognize it as a numerical entity. Additionally, it fails to tag "गाँधी" as part of the organization name "राजीव गाँधी" in the third sentence.
Overall, the IndicNER model demonstrates good performance in identifying and tagging named entities in the given text, but there is room for improvement in handling ambiguous cases or rare entities, as well as in correctly identifying entity boundaries within longer named entities.

PRECISION RECALL AND F SCORE OF COMPARISON BETWEEN MANUAL AND INDICBERT ANSWERS



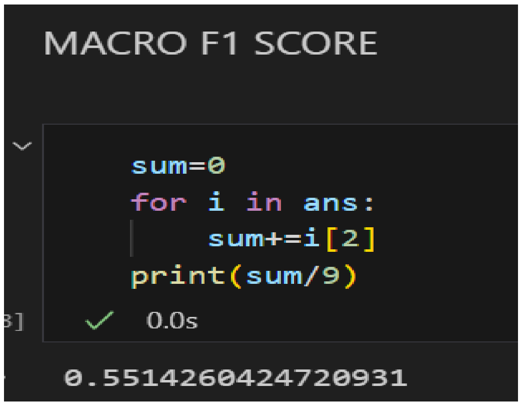
PRECISION RECALL AND FSCORE OF EACK TAG (MANUAL VS INDICNER)

```
print('Tags: (precision,recall,fscore)')
for i in range (0,9):
    print(Z[i],end=': ')
    print(ans[i])
```

0.0s

```
Tags: (precision,recall,fscore)
O: (0.9270462633451957, 0.9811676082862524, 0.9533394327538883)
B-MISC: (0, 0.0, 0)
I-MISC: (0, 0.0, 0)
B-PER: (0.5384615384615384, 0.5833333333333334, 0.5599999999999999)
I-PER: (0.5555555555555556, 0.5555555555555556, 0.5555555555555556)
B-ORG: (0.5555555555555556, 0.45454545454545453, 0.5)
I-ORG: (0.8, 0.6666666666666666, 0.7272727272727272)
B-LOC: (0.5, 1.0, 0.6666666666666666)
I-LOC: (1.0, 1.0, 1.0)
```

MACRO F1 SCORE OF COMPARISON BETWEEN MANUAL AND CHATGPT ANSWERS



MACRO F1 SCORE

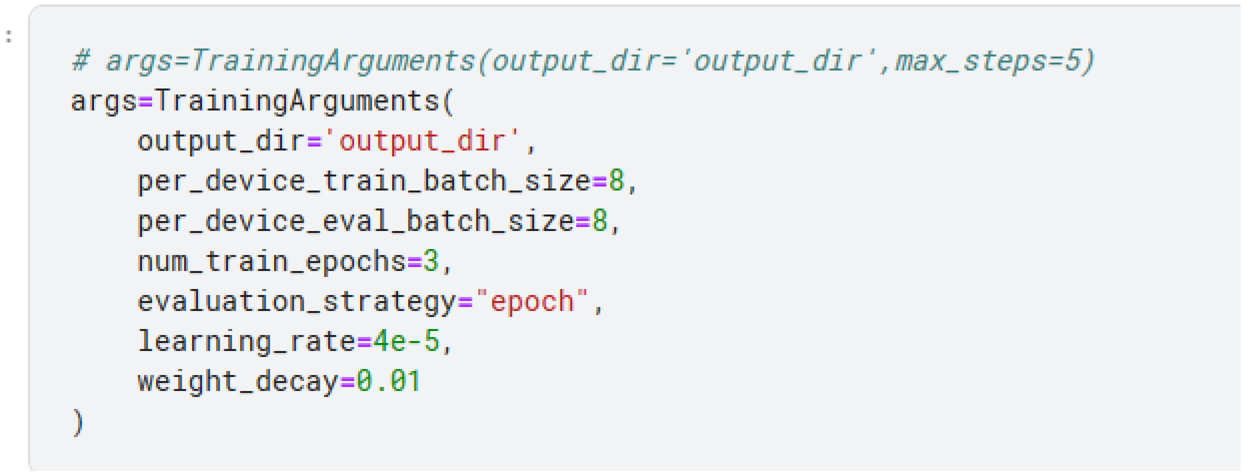
```
sum=0
for i in ans:
    sum+=i[2]
print(sum/9)
```

```
0.0s
```

```
0.5514260424720931
```

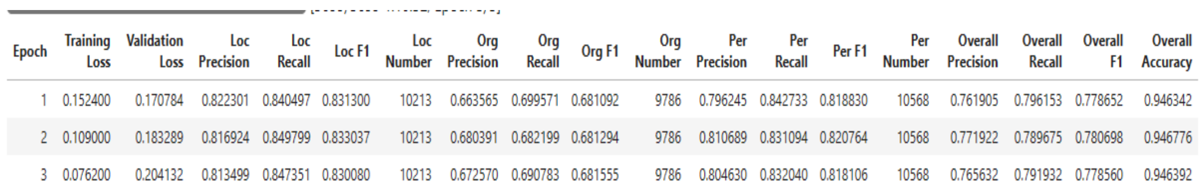
COMPARISON OF INDICBERT AND INDICNER ON Naamapadam CORPUS

PARAMETER USED:



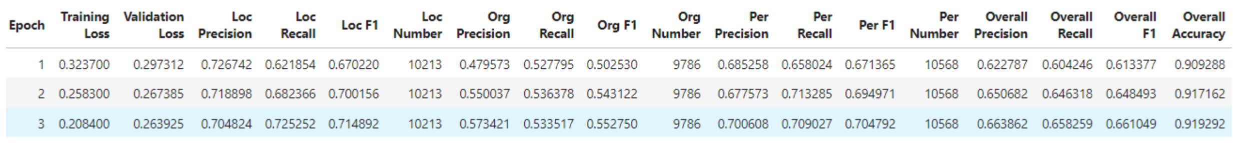
```
# args=TrainingArguments(output_dir='output_dir',max_steps=5)
args=TrainingArguments(
    output_dir='output_dir',
    per_device_train_batch_size=8,
    per_device_eval_batch_size=8,
    num_train_epochs=3,
    evaluation_strategy="epoch",
    learning_rate=4e-5,
    weight_decay=0.01
)
```

OUTPUT ON VALIDATION DATASET OF INDICNER MODEL:



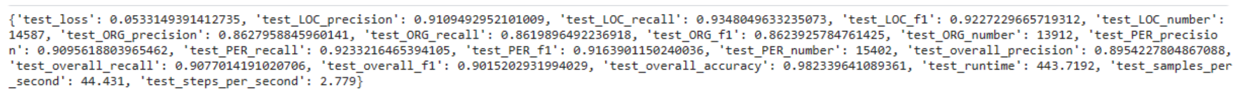
Epoch	Training Loss	Validation Loss	Loc Precision	Loc Recall	Loc F1	Loc Number	Org Precision	Org Recall	Org F1	Org Number	Per Precision	Per Recall	Per F1	Per Number	Overall Precision	Overall Recall	Overall F1	Overall Accuracy
1	0.152400	0.170784	0.822301	0.840497	0.831300	10213	0.663565	0.699571	0.681092	9786	0.796245	0.842733	0.818830	10568	0.761905	0.796153	0.778652	0.946342
2	0.109000	0.183289	0.816924	0.849799	0.833037	10213	0.680391	0.682199	0.681294	9786	0.810689	0.831094	0.820764	10568	0.771922	0.789675	0.780698	0.946776
3	0.076200	0.204132	0.813499	0.847351	0.830080	10213	0.672570	0.690783	0.681555	9786	0.804630	0.832040	0.818106	10568	0.765632	0.791932	0.778560	0.946392

OUTPUT ON VALIDATION DATASET OF INDICBERT MODEL:



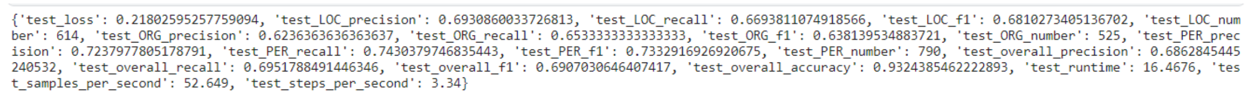
Epoch	Training Loss	Validation Loss	Loc Precision	Loc Recall	Loc F1	Loc Number	Org Precision	Org Recall	Org F1	Org Number	Per Precision	Per Recall	Per F1	Per Number	Overall Precision	Overall Recall	Overall F1	Overall Accuracy
1	0.323700	0.297312	0.726742	0.621854	0.670220	10213	0.479573	0.527795	0.502530	9786	0.685258	0.658024	0.671365	10568	0.622787	0.604246	0.613377	0.909288
2	0.258300	0.267385	0.718898	0.682366	0.700156	10213	0.550037	0.536378	0.543122	9786	0.677573	0.713285	0.694971	10568	0.650682	0.646318	0.648493	0.917162
3	0.208400	0.263925	0.704824	0.725252	0.714892	10213	0.573421	0.533517	0.552750	9786	0.700608	0.709027	0.704792	10568	0.663862	0.658259	0.661049	0.919292

OUTPUT ON TEST DATASET OF INDICNER MODEL:



```
{'test_loss': 0.0533149391412735, 'test_LOC_precision': 0.9109492952101009, 'test_LOC_recall': 0.9348049633235073, 'test_LOC_f1': 0.9227229665719312, 'test_LOC_number':
14587, 'test_ORG_precision': 0.8627958845960141, 'test_ORG_recall': 0.8619896492236918, 'test_ORG_f1': 0.8623925784761425, 'test_ORG_number': 13912, 'test_PER_precisio
n': 0.9095618803965462, 'test_PER_recall': 0.9233216465394105, 'test_PER_f1': 0.9163901150240036, 'test_PER_number': 15402, 'test_overall_precision': 0.8954227804867088,
'test_overall_recall': 0.907701419102070б, 'test_overall_f1': 0.9015202931994029, 'test_overall_accuracy': 0.982339641089361, 'test_runtime': 443.7192, 'test_samples_per
_second': 44.431, 'test_steps_per_second': 2.779}
```

OUTPUT ON TEST DATASET OF INDICBERT MODEL:



```
{'test_loss': 0.21802595257759094, 'test_LOC_precision': 0.6930860033726813, 'test_LOC_recall': 0.6693811074918566, 'test_LOC_f1': 0.6810273405136702, 'test_LOC_num
ber': 614, 'test_ORG_precision': 0.6236363636363637, 'test_ORG_recall': 0.6533333333333333, 'test_ORG_f1': 0.638139534883721, 'test_ORG_number': 525, 'test_PER_prec
ision': 0.7237977805178791, 'test_PER_recall': 0.7430379746835443, 'test_PER_f1': 0.7332916926920675, 'test_PER_number': 790, 'test_overall_precision': 0.6862845445
240532, 'test_overall_recall': 0.6951788491446346, 'test_overall_f1': 0.6907030646407417, 'test_overall_accuracy': 0.9324385462222893, 'test_runtime': 16.4676, 'tes
t_samples_per_second': 52.649, 'test_steps_per_second': 3.34}
```

CONCLUSION

In both the test and validation datasets, the performance of the INDICNER model surpasses that of other models in terms of overall accuracy. This indicates that the INDICNER model excels in accurately identifying and classifying named entities across various texts. Specifically, when evaluated on the Naamapadam corpus, the INDICNER model demonstrates superior performance compared to other models, showcasing its effectiveness and reliability in handling named entity recognition tasks.

LEARNING

Indeed, the same word can often carry different Named Entity Recognition (NER) tags depending on its context within a sentence. This variability arises from the fact that words can serve different linguistic functions and convey diverse meanings based on the surrounding words and the overall sentence structure.
For example, consider the word "Apple." In one context, it could refer to the fruit and be tagged as an entity under the category of "O" (ordinary word). In another context, it could refer to the multinational technology company and be tagged as an entity under the category of "ORG" (organization). Additionally, it could represent a person's name and be tagged as an entity under the category of "PER" (person) if it refers to someone with that name.

In the evaluation conducted on the Naamapadam Corpus, IndicNER outperformed IndicBERT when using specific hyperparameters: learning rate set to 4e-5, batch size of 8, and a word decay of 0.01. This indicates that, under these settings, IndicNER demonstrated superior performance in Named Entity Recognition tasks compared to IndicBERT. Such findings underscore the importance of fine-tuning hyperparameters for optimal performance in NLP tasks, particularly in domain-specific datasets like the Naamapadam Corpus.

IndicNER outperforms IndicBERT in Named Entity Recognition because it is specifically trained for this task, allowing it to better understand and recognize named entities in text data.

IndicNER's accuracy remains stable even with parameter variations, indicating robust performance across different settings.

We trained on 20000 line which is small dataset for training so accuracy is less especially in indicbert

Training on a small dataset of 20,000 lines results in lower accuracy, particularly for IndicBERT, due to limited exposure to varied linguistic patterns and contexts.