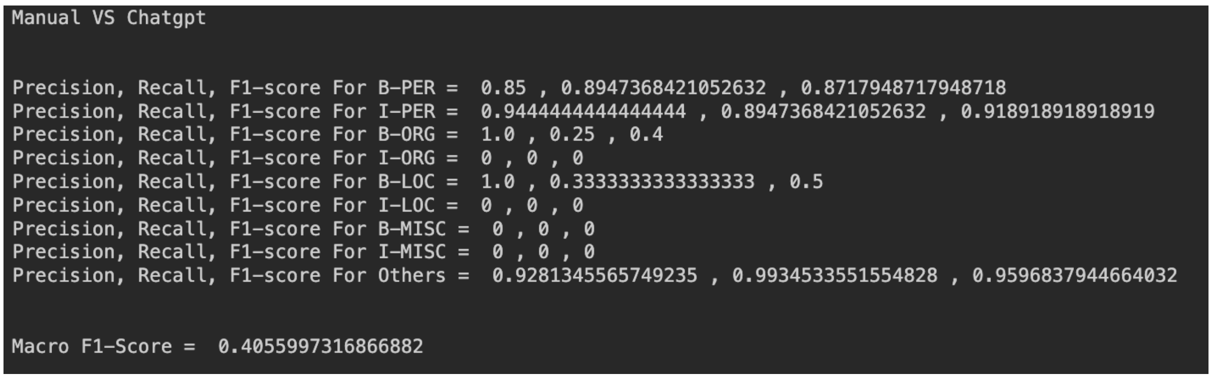
# Assignment 2

## Introduction

Named Entity Recognition (NER) is a fundamental task in natural language processing that involves identifying and classifying named entities in text into predefined categories such as persons, organisations, locations, and miscellaneous entities. In this report I have done a comparative analysis of NER tagging done by me manually and chatgpt as well as manual vs two models: Indic_BERT and Indic_ner. The evaluation metrics include precision, recall, and macro F1-score for each entity type, along with an overall macro F1-score.

## Comparison Results

### Manual VS Chatgpt



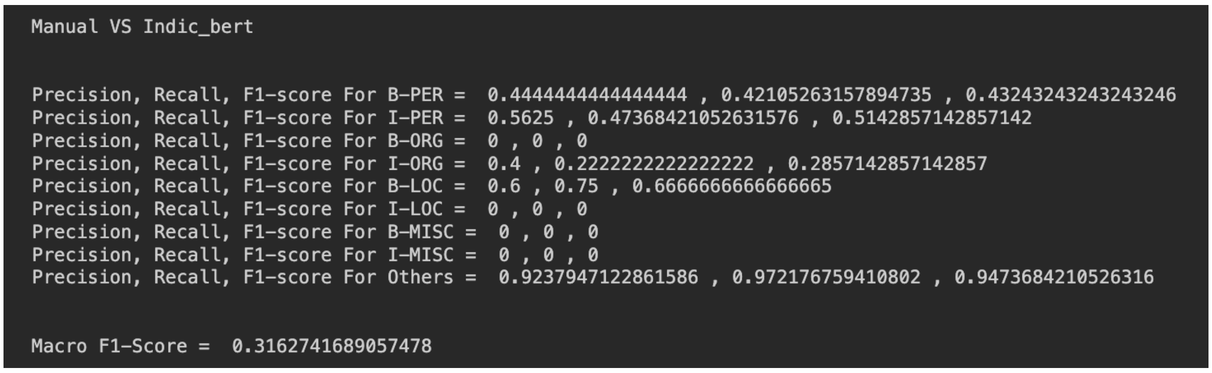
```
Manual VS Chatgpt


Precision, Recall, F1-score For B-PER =  0.85 , 0.8947368421052632 , 0.8717948717948718
Precision, Recall, F1-score For I-PER =  0.9444444444444444 , 0.8947368421052632 , 0.918918918918919
Precision, Recall, F1-score For B-ORG =  1.0 , 0.25 , 0.4
Precision, Recall, F1-score For I-ORG =  0 , 0 , 0
Precision, Recall, F1-score For B-LOC =  1.0 , 0.3333333333333333 , 0.5
Precision, Recall, F1-score For I-LOC =  0 , 0 , 0
Precision, Recall, F1-score For B-MISC =  0 , 0 , 0
Precision, Recall, F1-score For I-MISC =  0 , 0 , 0
Precision, Recall, F1-score For Others =  0.9281345565749235 , 0.9934533551554828 , 0.9596837944664032


Macro F1-Score =  0.4055997316866882
```

ChatGPT performed very well especially in identifying persons B-PER and I-PER classes with a precision of 0.85 and a recall of 0.89, resulting in a macro F1-score of 0.87 and precision of 0.94, recall of 0.89 and f1 score of 0.92 respectively. It also performs well on B-ORG and B-LOC classes.  However, its performance in recognizing organisations I-ORG and locations I-LOC is not good with zero precision and recall. ChatGPT shows good bad results in identifying miscellaneous entities with a precision of 0 and a recall of 0. For Others class it shows very good results with very high precision, recall and F-1 score. The macro F-1 score for chatgpt is 0.4.

## Manual vs Indic_BERT



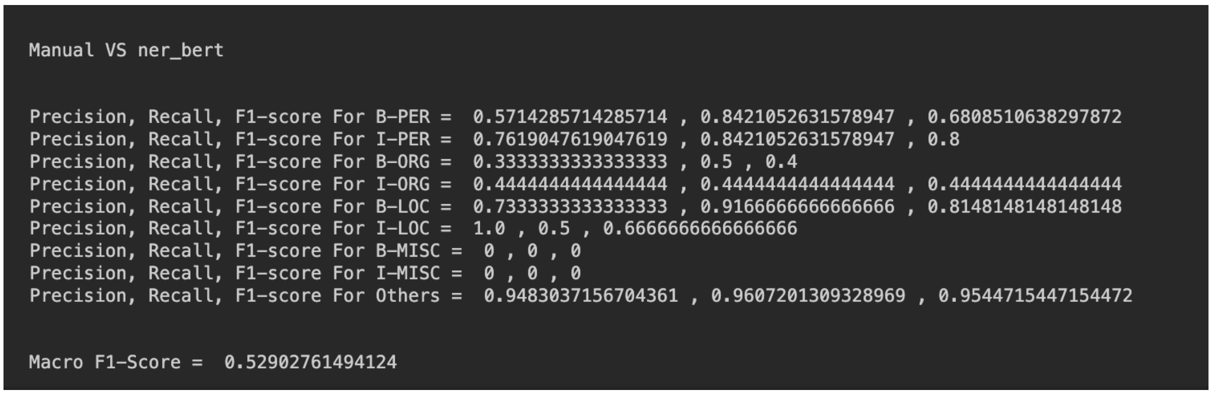
```
Manual VS Indic_bert


Precision, Recall, F1-score For B-PER =  0.4444444444444444 , 0.42105263157894735 , 0.43243243243243246
Precision, Recall, F1-score For I-PER =  0.5625 , 0.47368421052631576 , 0.5142857142857142
Precision, Recall, F1-score For B-ORG =  0 , 0 , 0
Precision, Recall, F1-score For I-ORG =  0.4 , 0.2222222222222222 , 0.2857142857142857
Precision, Recall, F1-score For B-LOC =  0.6 , 0.75 , 0.6666666666666665
Precision, Recall, F1-score For I-LOC =  0 , 0 , 0
Precision, Recall, F1-score For B-MISC =  0 , 0 , 0
Precision, Recall, F1-score For I-MISC =  0 , 0 , 0
Precision, Recall, F1-score For Others =  0.9237947122861586 , 0.972176759410802 , 0.9473684210526316


Macro F1-Score =  0.3162741689057478
```

Indic_BERT struggles to match the performance of the manual annotations, particularly in recognizing persons B-PER, I-PER and organisations (B-ORG and I-ORG), achieving lower precision, recall, and F1-scores. Although it shows a relatively better performance in recognizing locations B-LOC with a F-1 score of 0.67, recall of 0.75, it still falls short compared to the manual annotations. Indic_BERT demonstrates good performance in identifying other entities with a F1-score of 0.95. However the macro F-1 score of the Indic_BERT is 0.32 which is low compared to both other models.

## Manual vs Indic_ner



```
Manual VS ner_bert


Precision, Recall, F1-score For B-PER =  0.5714285714285714 , 0.8421052631578947 , 0.6808510638297872
Precision, Recall, F1-score For I-PER =  0.7619047619047619 , 0.8421052631578947 , 0.8
Precision, Recall, F1-score For B-ORG =  0.3333333333333333 , 0.5 , 0.4
Precision, Recall, F1-score For I-ORG =  0.4444444444444444 , 0.4444444444444444 , 0.4444444444444444
Precision, Recall, F1-score For B-LOC =  0.7333333333333333 , 0.9166666666666666 , 0.8148148148148148
Precision, Recall, F1-score For I-LOC =  1.0 , 0.5 , 0.6666666666666666
Precision, Recall, F1-score For B-MISC =  0 , 0 , 0
Precision, Recall, F1-score For I-MISC =  0 , 0 , 0
Precision, Recall, F1-score For Others =  0.9483037156704361 , 0.9607201309328969 , 0.9544715447154472


Macro F1-Score =  0.52902761494124
```

Indic_ner showcases improved performance compared to Indic_BERT as well as chatgpt across most categories. It excels in recognizing persons (B-PER and I-PER) and locations (B-LOC and I-LOC), achieving higher precision, recall, and F1-scores compared to Indic_BERT. However, it still struggles with identifying organisations (ORG) and MISC, although it shows a slight improvement over Indic_BERT. Indic_ner achieves the highest macro F1-score of 0.53 among the compared models.

## Hyperparameter Tuning

## Introduction:

Hyperparameters are critical entities that control the behaviour and performance of machine learning models. Tuning these hyperparameters appropriately can significantly impact the efficiency and effectiveness of model training and ultimately the performance of the resulting model. In this report, we delve into the significance of the hyperparameters tuned for fine-tuning the Indic BERT and Indic NER models for the named entity representation task. Additionally, we discuss the optimal values chosen based on experimentation and analysis.

### Hyperparameters Tuned

### 1. Batch Size (per_device_train_batch_size and per_device_eval_batch_size):

Significance: Batch size determines the number of samples processed before updating the model parameters during training. Larger batch sizes lead to faster convergence but may require more memory and computational resources. Smaller batch sizes may result in noisier gradients but allow for better generalisation.

Tested For values: 8, 10, 12

Optimal Value Chosen: 8

### 2. Number of Training Epochs (num_train_epochs):

Significance:The number of training epochs defines how many times the entire training dataset is passed through the model during training. Too few epochs may

result in underfitting, while too many epochs may lead to overfitting.

Value Chosen: 3

### 3. Evaluation Strategy (evaluation_strategy):

Significance: Determines when the model performance is evaluated during training. Evaluating the model at the end of each epoch allows monitoring performance and making decisions such as early stopping.

**4. Learning Rate (learning_rate):**

Significance: The learning rate controls the size of the step taken during optimization. A higher learning rate may result in faster convergence, but too high a learning rate may cause the model to overshoot optimal parameter values. Conversely, a lower learning rate may lead to slow convergence.

Tested for values: 1e-5, 2e-5, 4e-5

Optimal Value Chosen: 4e-5

**5. Weight Decay (weight_decay):**

Significance: Weight decay is a form of regularisation that penalises large parameter values during optimization. It helps prevent overfitting by discouraging overly complex models.

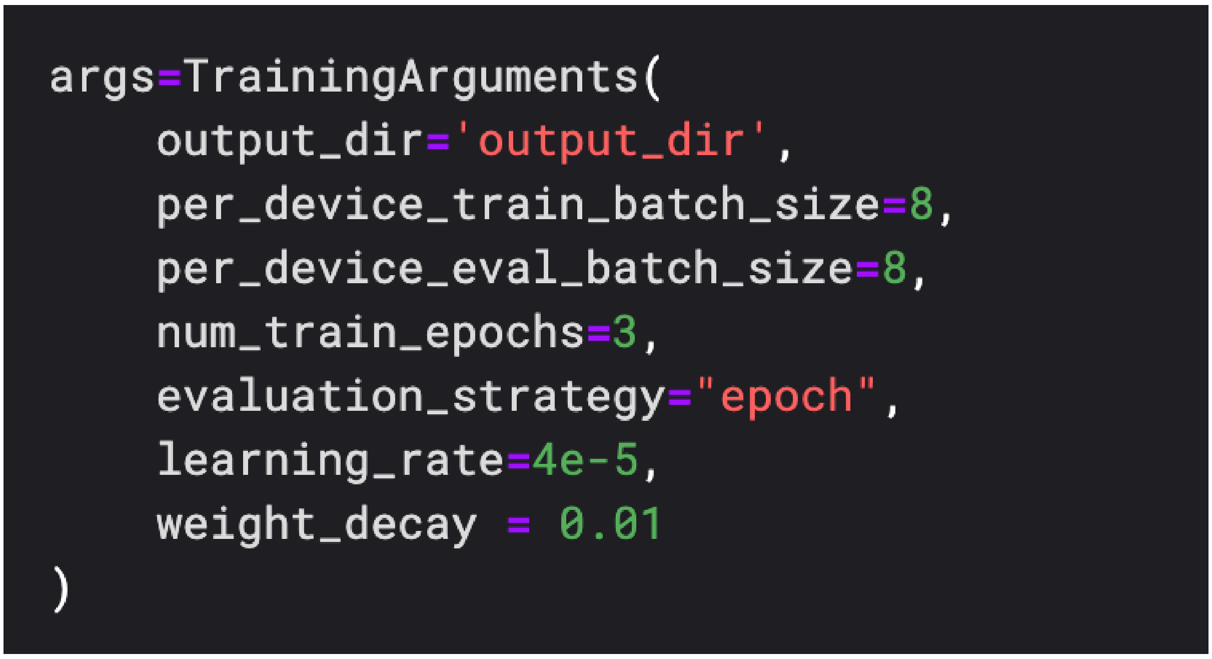
Tested for values: 0.01, 0.02

Optimal Value Chosen: 0.01

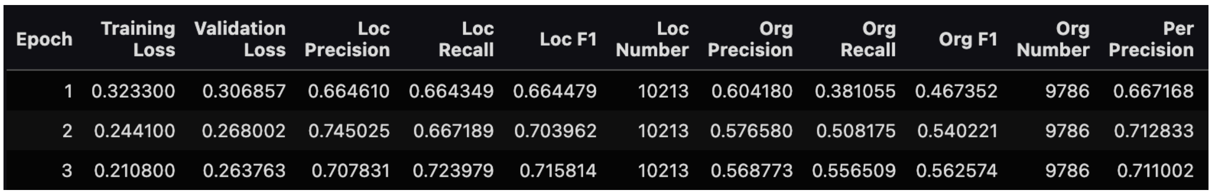
## OUTPUT

### Indic_bert

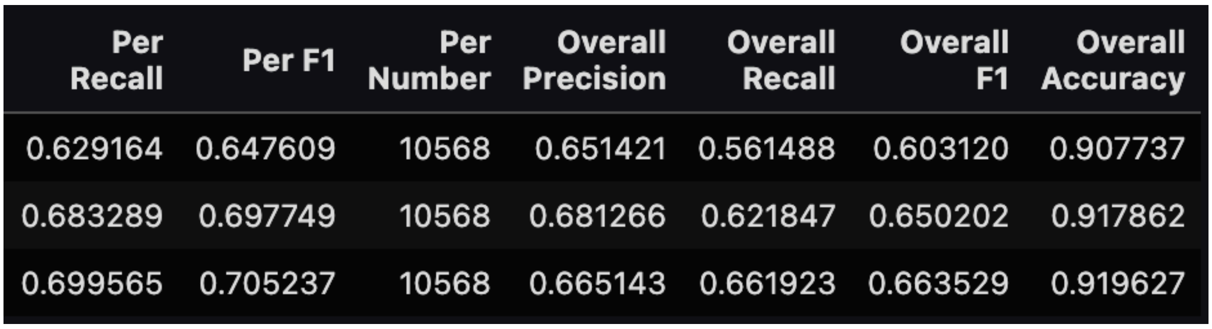
1



```
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
)
```

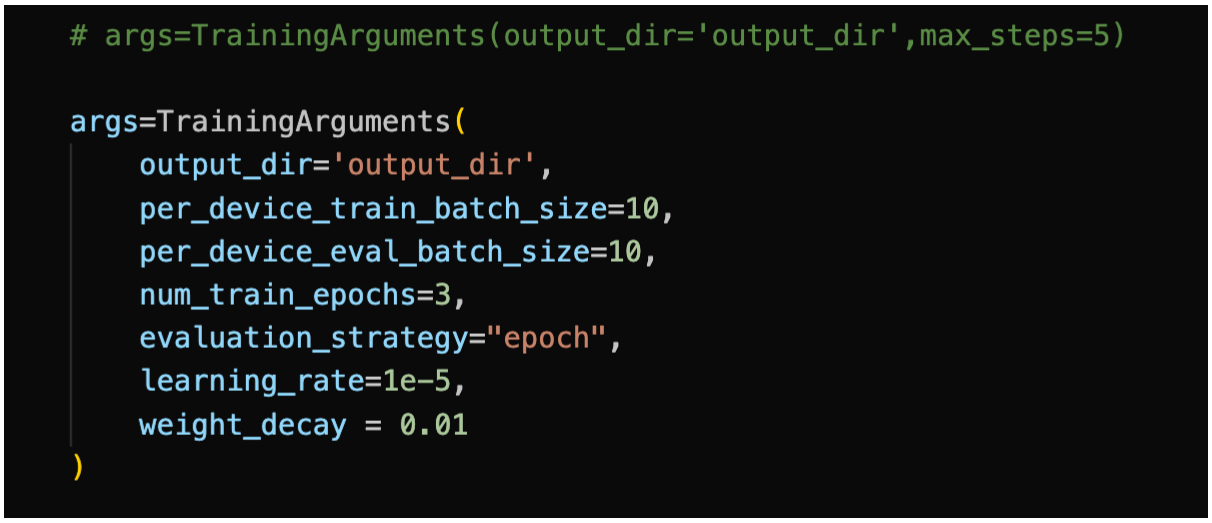


| Epoch | Training Loss | Validation Loss | Loc Precision | Loc Recall | Loc F1 | Loc Number | Org Precision | Org Recall | Org F1 | Org Number | Per Precision |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0.323300 | 0.306857 | 0.664610 | 0.664349 | 0.664479 | 10213 | 0.604180 | 0.381055 | 0.467352 | 9786 | 0.667168 |
| 2 | 0.244100 | 0.268002 | 0.745025 | 0.667189 | 0.703962 | 10213 | 0.576580 | 0.508175 | 0.540221 | 9786 | 0.712833 |
| 3 | 0.210800 | 0.263763 | 0.707831 | 0.723979 | 0.715814 | 10213 | 0.568773 | 0.556509 | 0.562574 | 9786 | 0.711002 |



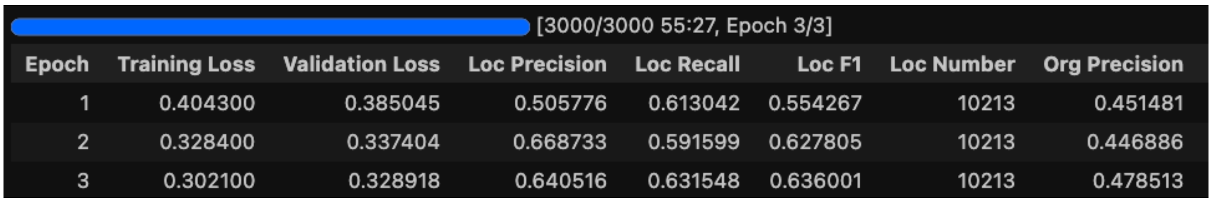
| Per Recall | Per F1 | Per Number | Overall Precision | Overall Recall | Overall F1 | Overall Accuracy |
|---|---|---|---|---|---|---|
| 0.629164 | 0.647609 | 10568 | 0.651421 | 0.561488 | 0.603120 | 0.907737 |
| 0.683289 | 0.697749 | 10568 | 0.681266 | 0.621847 | 0.650202 | 0.917862 |
| 0.699565 | 0.705237 | 10568 | 0.665143 | 0.661923 | 0.663529 | 0.919627 |

2



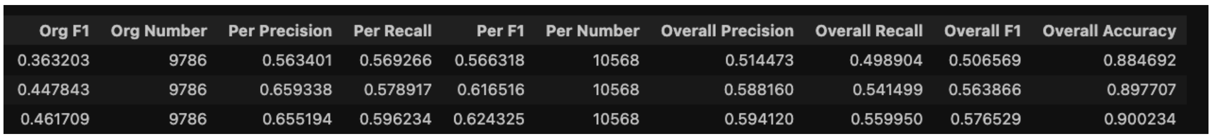
```
# args=TrainingArguments(output_dir='output_dir',max_steps=5)

args=TrainingArguments(
    output_dir='output_dir',
    per_device_train_batch_size=10,
    per_device_eval_batch_size=10,
    num_train_epochs=3,
    evaluation_strategy="epoch",
    learning_rate=1e-5,
    weight_decay = 0.01
)
```



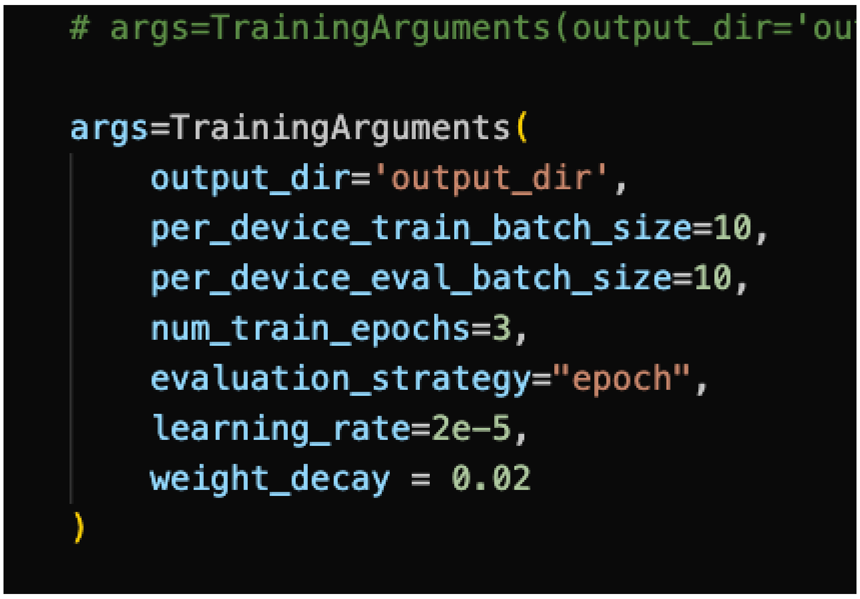
[3000/3000 55:27, Epoch 3/3]

| Epoch | Training Loss | Validation Loss | Loc Precision | Loc Recall | Loc F1 | Loc Number | Org Precision |
|---|---|---|---|---|---|---|---|
| 1 | 0.404300 | 0.385045 | 0.505776 | 0.613042 | 0.554267 | 10213 | 0.451481 |
| 2 | 0.328400 | 0.337404 | 0.668733 | 0.591599 | 0.627805 | 10213 | 0.446886 |
| 3 | 0.302100 | 0.328918 | 0.640516 | 0.631548 | 0.636001 | 10213 | 0.478513 |



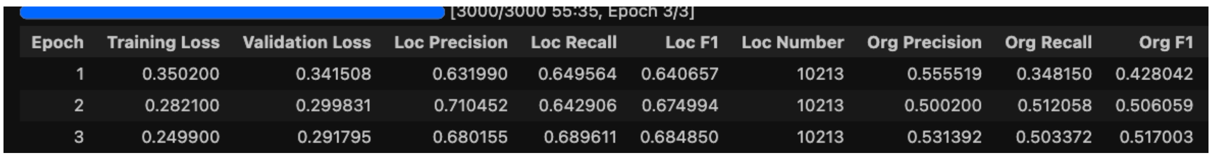
| Org F1 | Org Number | Per Precision | Per Recall | Per F1 | Per Number | Overall Precision | Overall Recall | Overall F1 | Overall Accuracy |
|---|---|---|---|---|---|---|---|---|---|
| 0.363203 | 9786 | 0.563401 | 0.569266 | 0.566318 | 10568 | 0.514473 | 0.498904 | 0.506569 | 0.884692 |
| 0.447843 | 9786 | 0.659338 | 0.578917 | 0.616516 | 10568 | 0.588160 | 0.541499 | 0.563866 | 0.897707 |
| 0.461709 | 9786 | 0.655194 | 0.596234 | 0.624325 | 10568 | 0.594120 | 0.559950 | 0.576529 | 0.900234 |

3

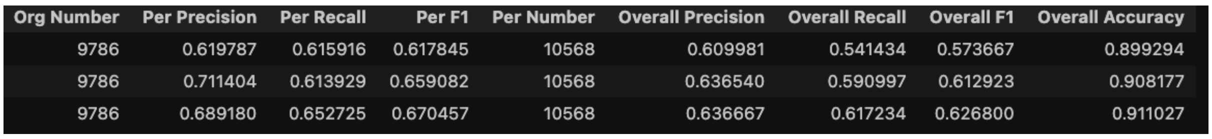


```
# args=TrainingArguments(output_dir='ou

args=TrainingArguments(
    output_dir='output_dir',
    per_device_train_batch_size=10,
    per_device_eval_batch_size=10,
    num_train_epochs=3,
    evaluation_strategy="epoch",
    learning_rate=2e-5,
    weight_decay = 0.02
)
```


[3000/3000 55:35, Epoch 3/3]

| Epoch | Training Loss | Validation Loss | Loc Precision | Loc Recall | Loc F1 | Loc Number | Org Precision | Org Recall | Org F1 |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 0.350200 | 0.341508 | 0.631990 | 0.649564 | 0.640657 | 10213 | 0.555519 | 0.348150 | 0.428042 |
| 2 | 0.282100 | 0.299831 | 0.710452 | 0.642906 | 0.674994 | 10213 | 0.500200 | 0.512058 | 0.506059 |
| 3 | 0.249900 | 0.291795 | 0.680155 | 0.689611 | 0.684850 | 10213 | 0.531392 | 0.503372 | 0.517003 |



| Org Number | Per Precision | Per Recall | Per F1 | Per Number | Overall Precision | Overall Recall | Overall F1 | Overall Accuracy |
|---|---|---|---|---|---|---|---|---|
| 9786 | 0.619787 | 0.615916 | 0.617845 | 10568 | 0.609981 | 0.541434 | 0.573667 | 0.899294 |
| 9786 | 0.711404 | 0.613929 | 0.659082 | 10568 | 0.636540 | 0.590997 | 0.612923 | 0.908177 |
| 9786 | 0.689180 | 0.652725 | 0.670457 | 10568 | 0.636667 | 0.617234 | 0.626800 | 0.911027 |

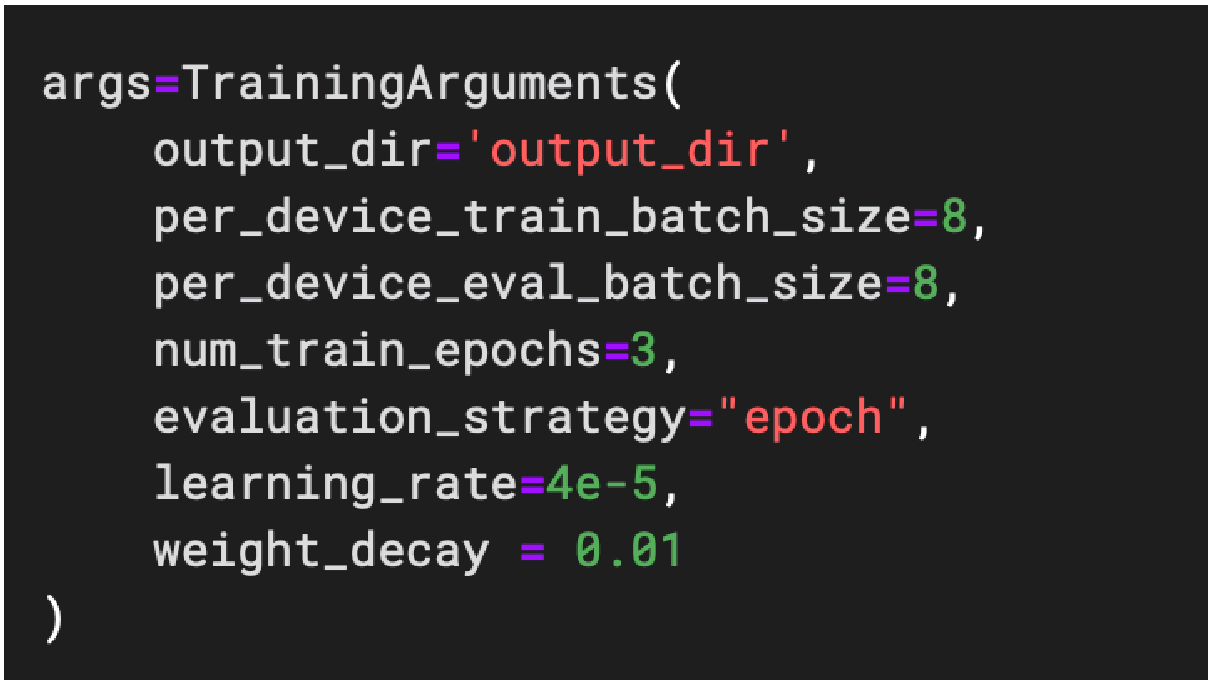
**Optimum Values of Hyperparameters:**

Batch size = 8
Learning rate = 4e-5
Weight decay = 0.01

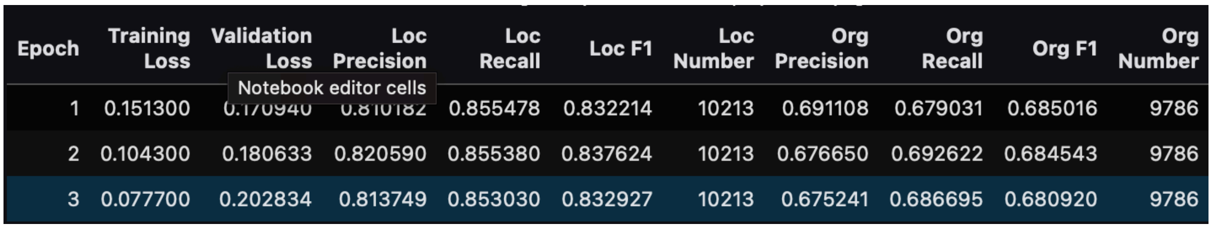
**Macro F-1 score** : For testing - 0.6827166151844554
For training - 0.7474777890135678
For validation - 0.663529

**Indic_ner**

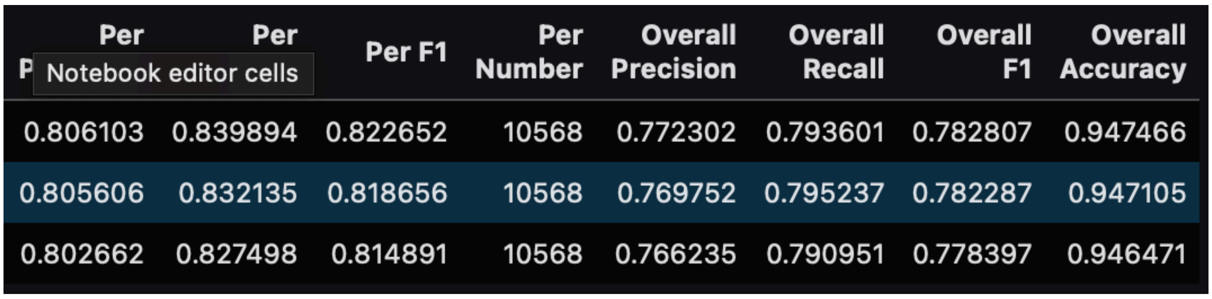
1



```
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
)
```

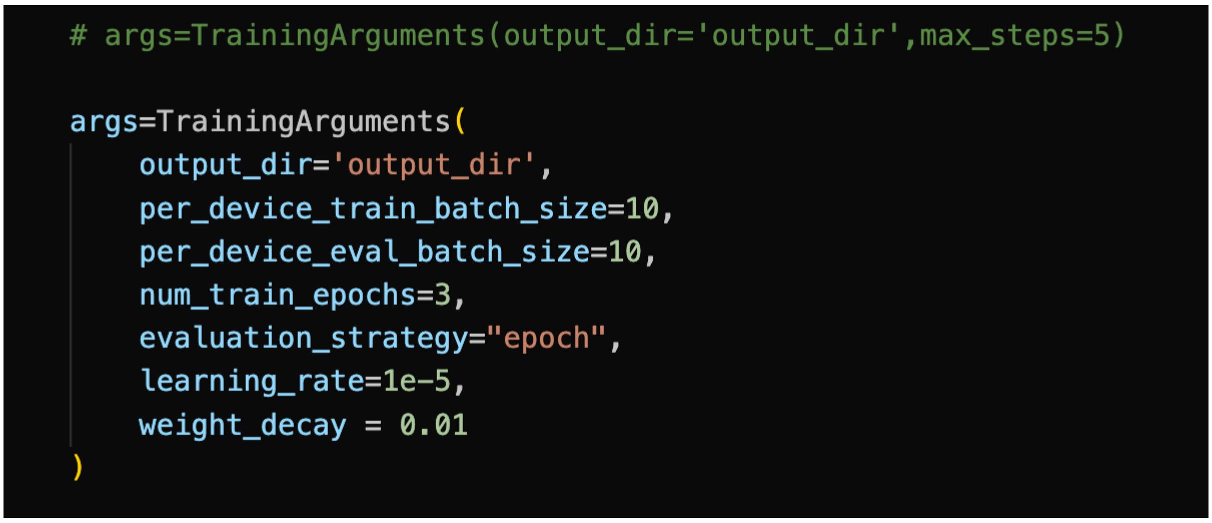

Notebook editor cells

| Epoch | Training Loss | Validation Loss | Loc Precision | Loc Recall | Loc F1 | Loc Number | Org Precision | Org Recall | Org F1 | Org Number |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0.151300 | 0.170940 | 0.810182 | 0.855478 | 0.832214 | 10213 | 0.691108 | 0.679031 | 0.685016 | 9786 |
| 2 | 0.104300 | 0.180633 | 0.820590 | 0.855380 | 0.837624 | 10213 | 0.676650 | 0.692622 | 0.684543 | 9786 |
| 3 | 0.077700 | 0.202834 | 0.813749 | 0.853030 | 0.832927 | 10213 | 0.675241 | 0.686695 | 0.680920 | 9786 |


Notebook editor cells

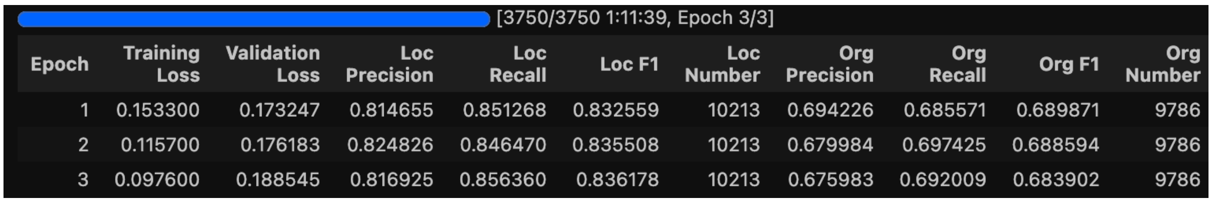
| Per P | Per | Per F1 | Per Number | Overall Precision | Overall Recall | Overall F1 | Overall Accuracy |
|---|---|---|---|---|---|---|---|
| 0.806103 | 0.839894 | 0.822652 | 10568 | 0.772302 | 0.793601 | 0.782807 | 0.947466 |
| 0.805606 | 0.832135 | 0.818656 | 10568 | 0.769752 | 0.795237 | 0.782287 | 0.947105 |
| 0.802662 | 0.827498 | 0.814891 | 10568 | 0.766235 | 0.790951 | 0.778397 | 0.946471 |

2



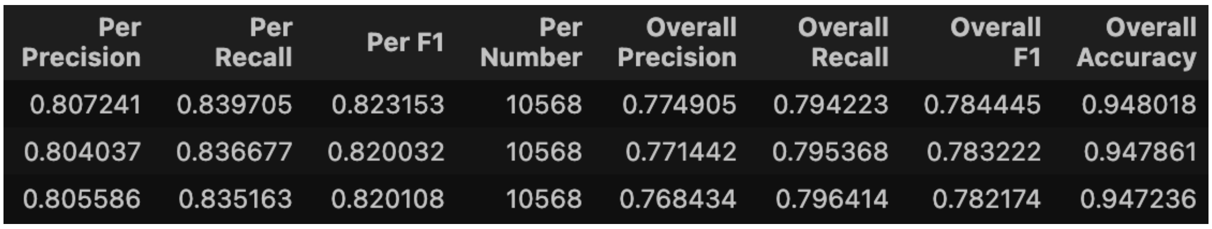
```
# args=TrainingArguments(output_dir='output_dir',max_steps=5)

args=TrainingArguments(
    output_dir='output_dir',
    per_device_train_batch_size=10,
    per_device_eval_batch_size=10,
    num_train_epochs=3,
    evaluation_strategy="epoch",
    learning_rate=1e-5,
    weight_decay = 0.01
)
```



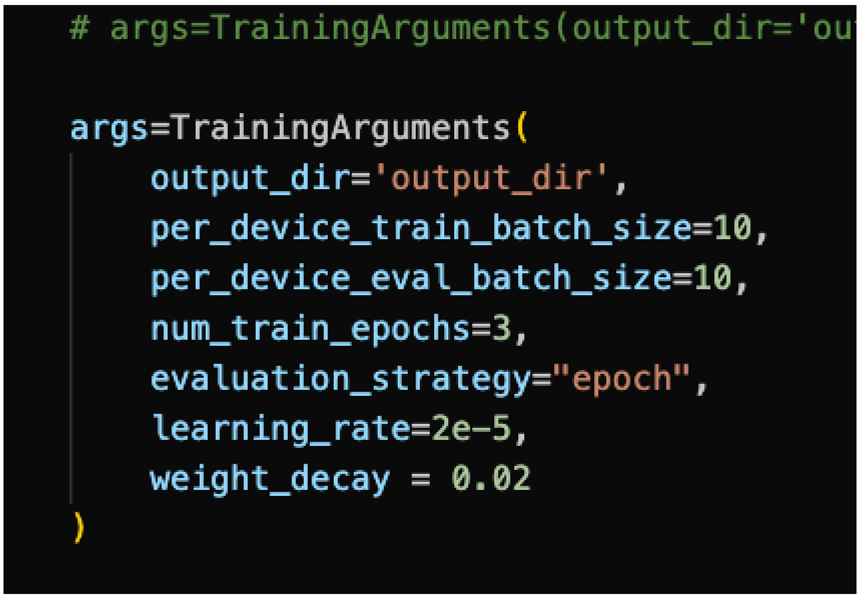
[3750/3750 1:11:39, Epoch 3/3]

| Epoch | Training Loss | Validation Loss | Loc Precision | Loc Recall | Loc F1 | Loc Number | Org Precision | Org Recall | Org F1 | Org Number |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0.153300 | 0.173247 | 0.814655 | 0.851268 | 0.832559 | 10213 | 0.694226 | 0.685571 | 0.689871 | 9786 |
| 2 | 0.115700 | 0.176183 | 0.824826 | 0.846470 | 0.835508 | 10213 | 0.679984 | 0.697425 | 0.688594 | 9786 |
| 3 | 0.097600 | 0.188545 | 0.816925 | 0.856360 | 0.836178 | 10213 | 0.675983 | 0.692009 | 0.683902 | 9786 |



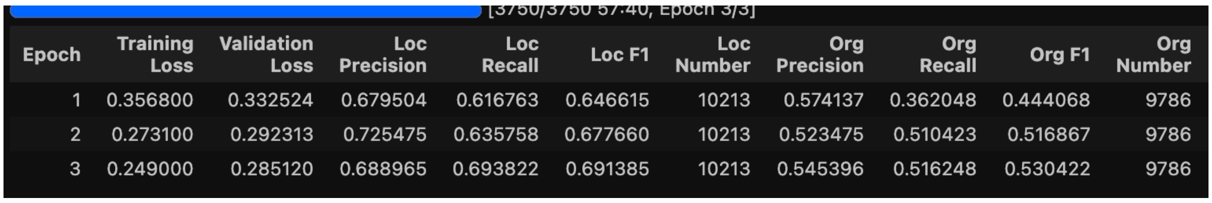
| Per Precision | Per Recall | Per F1 | Per Number | Overall Precision | Overall Recall | Overall F1 | Overall Accuracy |
|---|---|---|---|---|---|---|---|
| 0.807241 | 0.839705 | 0.823153 | 10568 | 0.774905 | 0.794223 | 0.784445 | 0.948018 |
| 0.804037 | 0.836677 | 0.820032 | 10568 | 0.771442 | 0.795368 | 0.783222 | 0.947861 |
| 0.805586 | 0.835163 | 0.820108 | 10568 | 0.768434 | 0.796414 | 0.782174 | 0.947236 |

3



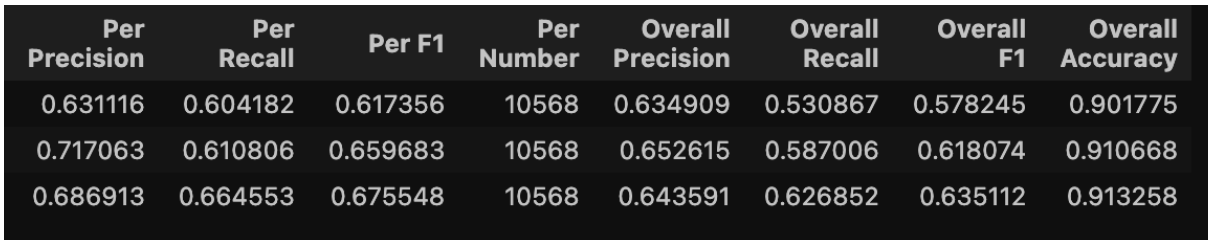
```
# args=TrainingArguments(output_dir='ou

args=TrainingArguments(
    output_dir='output_dir',
    per_device_train_batch_size=10,
    per_device_eval_batch_size=10,
    num_train_epochs=3,
    evaluation_strategy="epoch",
    learning_rate=2e-5,
    weight_decay = 0.02
)
```



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

| Epoch | Training Loss | Validation Loss | Loc Precision | Loc Recall | Loc F1 | Loc Number | Org Precision | Org Recall | Org F1 | Org Number |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0.356800 | 0.332524 | 0.679504 | 0.616763 | 0.646615 | 10213 | 0.574137 | 0.362048 | 0.444068 | 9786 |
| 2 | 0.273100 | 0.292313 | 0.725475 | 0.635758 | 0.677660 | 10213 | 0.523475 | 0.510423 | 0.516867 | 9786 |
| 3 | 0.249000 | 0.285120 | 0.688965 | 0.693822 | 0.691385 | 10213 | 0.545396 | 0.516248 | 0.530422 | 9786 |



| Per Precision | Per Recall | Per F1 | Per Number | Overall Precision | Overall Recall | Overall F1 | Overall Accuracy |
|---|---|---|---|---|---|---|---|
| 0.631116 | 0.604182 | 0.617356 | 10568 | 0.634909 | 0.530867 | 0.578245 | 0.901775 |
| 0.717063 | 0.610806 | 0.659683 | 10568 | 0.652615 | 0.587006 | 0.618074 | 0.910668 |
| 0.686913 | 0.664553 | 0.675548 | 10568 | 0.643591 | 0.626852 | 0.635112 | 0.913258 |

**Optimum Values of Hyperparameters:**

Batch size = 8
Learning rate = 4e-5
Weight decay = 0.01

**Macro F-1 score** : For testing - 0.7949344470194554
For training - 0.9010216319553904
For validation - 0.778397

## Learnings

From this assignment, we learn the importance of named entity recognition (NER) in natural language processing tasks, especially for Indian languages. By manually annotating sentences and fine-tuning pre-trained models such as IndicBERT and IndicNER on a specialised corpus, we gain insights into the effectiveness of different models in capturing named entities accurately. Furthermore, leveraging state-of-the-art language models like ChatGPT for new tasks highlights the versatility of such models in handling various NLP tasks. By evaluating the performance of these models using precision, recall, and macro-F1 scores, we understand the trade-offs and limitations of each approach. Finally, the process of tuning hyperparameters, such as batch size, learning rate, and weight decay, underscores the importance of optimization for achieving optimal model performance while preventing overfitting.

## Conclusion

In conclusion, while the manual annotations serve as the standard, both ChatGPT and BERT-based models offer viable alternatives for named entity recognition tasks. ChatGPT shows strong performance in identifying persons and miscellaneous entities but falls short in recognizing organisations and locations. On the other hand, Indic_ner outperforms Indic_BERT across most categories and achieves the highest overall performance among the compared models.

In this report, detailed insights were provided into the significance of the hyperparameters tuned for fine-tuning the Indic BERT and Indic NER models. By selecting optimal values for batch size, learning rate, and weight decay, the aim was to achieve efficient training while preventing overfitting. The chosen hyperparameters have a balance between training efficiency, model regularisation, and performance on the named entity representation task. Further experimentation and tuning may be required to optimise the model further for specific datasets and tasks.