

# Assignment 2: CS689

Abinash Choudhary, Roll Number: 231110004

## 1 Comparing Models

### 1.1 IndicNER vs Manually Tagged NER Tags

IndicNER demonstrated strong performance compared to manually tagged NER tags. It achieved a precision of 0.930865, indicating that over 93% of its identified entities were correctly classified. The recall of 0.927405 highlights that IndicNER successfully found most of the relevant entities present in the data. These scores combine into an F1-score of 0.958058, which is considered excellent for NER tasks. While the macro F1-score of 0.516001 might seem lower, it's important to consider potential class imbalances in the dataset when interpreting this metric.

Results: Precision: 0.9308647652834692 Recall: 0.9274047186932849 F1: 0.9580580464240267 Macro F1: 0.5160007297938333

### 1.2 IndicBERT vs Manually Tagged NER Tags

IndicBERT exhibited commendable performance when compared against manually tagged NER tags. It attained a precision score of 0.847912, demonstrating that approximately 85% of the entities it identified were correct. The recall score of 0.868110 highlights IndicBERT's ability to locate a significant portion of the relevant entities within the data. These scores culminate in an F1-score of 0.922592, indicating strong overall performance in the NER task. Although the macro F1-score of 0.548039 may appear lower, it's important to consider potential class imbalances in the dataset when interpreting this metric.

Precision: 0.8479116740842176 Recall: 0.8681102362204725 F1: 0.922591599940071 Macro F1: 0.5480385247827109

### 1.3 ChatGPT vs Manually Tagged NER Tags

While ChatGPT demonstrated reasonable performance on the NER task, it exhibited some limitations compared to specialized models. Its precision of 0.843402 suggests that roughly 84% of its identified entities were correctly classified. The recall of 0.871143 indicates it found a good portion of relevant entities. These scores result in an overall F1-score of 0.932268, which is respectable. However, the macro F1-score of 0.770227 highlights potential challenges with certain entity classes. This may reflect ChatGPT's broader language focus, as opposed to the specialized training that IndicBERT and IndicNER received for NER.

Precision: 0.8434016074669433 Recall: 0.8711433756805808 F1: 0.9322678977322785 Macro F1: 0.7702265372168284

## 2 Hyperparameter Tuning

### 2.1 Number of Epochs:

The number of epochs, which represents the number of complete passes through the training dataset, is a crucial hyperparameter. I selected a value of 3 for several reasons. Firstly, this choice aimed to mitigate overfitting, an issue indicated by an increasing validation loss. Secondly, it allowed for reasonable convergence within the constraints of the 30,000-sentence dataset.

### 2.2 Per Device Train Batch Size:

This hyperparameter governs the number of training samples processed during each model update step. I opted for a batch size of 16. This value aimed to strike a balance between training speed and memory efficiency. A smaller batch size would likely decelerate training, while a larger batch size could potentially lead to out-of-memory errors.

### **2.3 Per Device Eval Batch Size:**

Similar to the train batch size, this parameter determines the number of samples used during evaluation phases. I selected a batch size of 16 for evaluation. This choice, like its training counterpart, prioritized a balance between evaluation speed and memory constraints.

### **2.4 Evaluation Strategy and Save Strategy:**

I employed a step-based evaluation strategy with evaluations occurring every 500 steps. This approach provided frequent insights into model performance during training. Additionally, to preserve the top-performing models, I implemented a strategy that saved the two best models based on their performance metrics.

### **2.5 Rationale for Hyperparameter Selection**

The selection of these hyperparameter values was guided by a combination of theoretical principles, practical constraints, and empirical observations. The primary objectives were to optimize model performance, prevent overfitting, and ensure computational feasibility within the given hardware limitations. Further experimentation and refinement may yield even greater improvements.

### 3 Results

#### 3.1 IndicNER with 30,000 sentences and 3 epochs




[5625/5625 25:55, Epoch 3/3]

Step	Training Loss	Validation Loss	Macro F1	Precision	Recall	Accuracy
500	0.406500	0.241796	0.828687	0.849053	0.810782	0.925989
1000	0.224400	0.226584	0.834325	0.835101	0.834304	0.927640
1500	0.220200	0.232453	0.833206	0.834716	0.833332	0.927621
2000	0.203100	0.240768	0.835622	0.836832	0.835491	0.927978
2500	0.156600	0.236515	0.835081	0.838254	0.832803	0.927876
3000	0.152300	0.235686	0.834562	0.839855	0.830177	0.928058
3500	0.158900	0.236911	0.835954	0.842905	0.830523	0.928282
4000	0.134200	0.269422	0.833161	0.841959	0.825169	0.927172
4500	0.112900	0.261736	0.831564	0.836732	0.827208	0.926547
5000	0.110300	0.264448	0.836010	0.838965	0.833401	0.927961
5500	0.111500	0.262278	0.834728	0.838729	0.831282	0.927510

Figure 1: IndicNER fine-tuned Model

IndicNER fine-tuned Model On test data

```
trainer.evaluate(testData)
```




[55/55 00:02]

```
{'eval_loss': 0.2256152629852295,  
'eval_macro_f1': 0.8433462008091087,  
'eval_precision': 0.8459805228289533,  
'eval_recall': 0.8469770832630077,  
'eval_accuracy': 0.9347412151635797,  
'eval_runtime': 2.9443,  
'eval_samples_per_second': 294.466,  
'eval_steps_per_second': 18.68,  
'epoch': 3.0}
```

Figure 2: IndicNER fine-tuned Model On test data

### 3.2 IndicBERT with 30,000 sentences and 3 epochs




[5625/5625 22:58, Epoch 3/3]

Step	Training Loss	Validation Loss	Macro F1	Precision	Recall	Accuracy
500	0.882600	0.755238	0.294485	0.723430	0.250022	0.768237
1000	0.660300	0.615878	0.459078	0.688088	0.420878	0.808318
1500	0.553900	0.513221	0.581776	0.704554	0.531885	0.839059
2000	0.491100	0.478587	0.619992	0.680979	0.597122	0.847821
2500	0.434400	0.444154	0.651283	0.709638	0.613842	0.858706
3000	0.417500	0.434524	0.657169	0.710247	0.626356	0.860676
3500	0.413700	0.415442	0.679788	0.716113	0.656855	0.867042
4000	0.371100	0.409678	0.690059	0.699959	0.682018	0.868382
4500	0.335300	0.402828	0.694747	0.709027	0.682794	0.870546
5000	0.325300	0.399252	0.694591	0.749947	0.655638	0.875379
5500	0.319600	0.393762	0.698374	0.743572	0.664799	0.876059

Figure 3: IndicBERT fine-tuned Model

IndicBert fine-tuned Model On test data

```
trainer_bert.evaluate(testData)
```



[55/55 00:02]

```
{'eval_loss': 0.3567679524421692,  
'eval_macro_f1': 0.7133476792867718,  
'eval_precision': 0.7503896223567134,  
'eval_recall': 0.6848482136519672,  
'eval_accuracy': 0.8858692516300272,  
'eval_runtime': 2.9862,  
'eval_samples_per_second': 290.333,  
'eval_steps_per_second': 18.418,  
'epoch': 3.0}
```

Figure 4: IndicBert fine-tuned Model On test data