Report on Assignment 2 on CS689A : Computational Linguistics for Indian Languages

By

Yashvir Singh Nathawat(231110059 M.tech CSE)

1. Comparison of Both Models on Namapadam Corpus - Hindi

On Namapadam dataset , IndicNER performs better than IndicBert as shown in below pictures.

Some Comparison between IndicNer and IndicBert based on below pictures:

- IndicNER shows better performance compared to IndicBert
- **Specific to Indian Languages :** IndicNER models are specially fine-tuned for Indian Languages, ensuring better understanding and handling nuances for morphologically rich languages like Hindi,Tamil,etc.

IndicBert During Training:

[7500/7500 58:01, Epoch 3/3]

Step	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
500	0.320100	0.346491	0.708628	0.539606	0.612674	10213	0.482123	0.398222	0.436174	9786	0.652571	0.575322	0.611516	10568	0.615165	0.506690	0.555683	0.897141
1000	0.324200	0.318100	0.702113	0.598649	0.646266	10213	0.503783	0.442264	0.471024	9786	0.706839	0.542771	0.614034	10568	0.636578	0.529264	0.577982	0.902196
1500	0.305700	0.293586	0.672130	0.682659	0.677354	10213	0.550400	0.471490	0.507898	9786	0.665309	0.664743	0.665026	10568	0.634863	0.608859	0.621589	0.910497
2000	0.279400	0.286097	0.697576	0.707138	0.702324	10213	0.546242	0.515430	0.530389	9786	0.719745	0.630867	0.672382	10568	0.656256	0.619393	0.637292	0.913340
2500	0.279000	0.283199	0.740949	0.645256	0.689800	10213	0.598902	0.434703	0.503760	9786	0.681157	0.655375	0.668017	10568	0.679152	0.581346	0.626454	0.913304
3000	0.234100	0.273426	0.719460	0.694311	0.706662	10213	0.535803	0.551298	0.543440	9786	0.690946	0.705526	0.698160	10568	0.649238	0.652403	0.650817	0.915677
3500	0.237900	0.269328	0.722036	0.716734	0.719375	10213	0.588584	0.507868	0.545255	9786	0.691003	0.709311	0.700037	10568	0.672307	0.647299	0.659566	0.918582
4000	0.223700	0.266063	0.674947	0.752864	0.711780	10213	0.597890	0.521153	0.556890	9786	0.743174	0.672218	0.705917	10568	0.674774	0.650800	0.662570	0.919670
4500	0.220800	0.257876	0.716366	0.732008	0.724103	10213	0.562789	0.571531	0.567126	9786	0.725420	0.714232	0.719783	10568	0.669840	0.674486	0.672155	0.921668
5000	0.219600	0.250990	0.724527	0.742191	0.733253	10213	0.593237	0.557531	0.574830	9786	0.741360	0.704391	0.722403	10568	0.689562	0.670004	0.679642	0.922897
5500	0.183900	0.256628	0.710708	0.755801	0.732561	10213	0.592783	0.553955	0.572711	9786	0.712537	0.721707	0.717093	10568	0.676229	0.679393	0.677807	0.922246
6000	0.178900	0.255390	0.740675	0.732987	0.736811	10213	0.576695	0.567443	0.572032	9786	0.723975	0.725208	0.724591	10568	0.682772	0.677299	0.680024	0.922539
6500	0.182000	0.256375	0.725620	0.742387	0.733908	10213	0.591520	0.561721	0.576236	9786	0.727614	0.715083	0.721294	10568	0.684944	0.675107	0.679990	0.923157
7000	0.178500	0.256834	0.721212	0.753060	0.736792	10213	0.586482	0.556816	0.571264	9786	0.716632	0.726533	0.721549	10568	0.678796	0.681061	0.679927	0.923650
7500	0.164800	0.255738	0.737568	0.743562	0.740553	10213	0.578472	0.569487	0.573944	9786	0.728235	0.722653	0.725433	10568	0.683960	0.680603	0.682277	0.923679

Accuracy = 92.36%

F1_Score = 68.22%

IndicBert Evaluation :

```
.]:
# Evaluating MODel
Final_metrics = trainer.evaluate()
trainer.log_metrics("eval", Final_metrics)
```

[1683/1683 01:58]

```
***** eval metrics *****
  epoch                   =        3.0
  eval_LOC_f1             =     0.7406
  eval_LOC_number         =      10213
  eval_LOC_precision      =     0.7376
  eval_LOC_recall         =     0.7436
  eval_ORG_f1             =     0.5739
  eval_ORG_number         =       9786
  eval_ORG_precision      =     0.5785
  eval_ORG_recall         =     0.5695
  eval_PER_f1             =     0.7254
  eval_PER_number         =      10568
  eval_PER_precision      =     0.7282
  eval_PER_recall         =     0.7227
  eval_loss               =     0.2557
  eval_overall_accuracy   =     0.9237
  eval_overall_f1         =     0.6823
  eval_overall_precision  =      0.684
  eval_overall_recall     =     0.6806
  eval_runtime            = 0:02:11.01
  eval_samples_per_second =    102.738
  eval_steps_per_second   =     12.846
```

+ Code + Markdown

Macro-F1 Score = 0.6823

IndicBert on Train Dataset

```
***** train metrics *****
  test_LOC_f1               =      0.832
  test_LOC_number           =      14841
  test_LOC_precision        =     0.8228
  test_LOC_recall           =     0.8413
  test_ORG_f1               =     0.7059
  test_ORG_number           =      14082
  test_ORG_precision        =     0.7151
  test_ORG_recall           =      0.697
  test_PER_f1               =     0.8359
  test_PER_number           =      15614
  test_PER_precision        =     0.8364
  test_PER_recall           =     0.8354
  test_loss                 =     0.1506
  test_overall_accuracy     =      0.955
  test_overall_f1           =      0.794
  test_overall_precision    =     0.7943
  test_overall_recall       =     0.7936
  test_runtime              = 0:03:15.09
  test_samples_per_second   =    102.516
  test_steps_per_second     =     12.814

 Macro f1 score:: 0.791260325081025
```

Indic Bert on 20% Test Dataset

```
***** Test metrics *****
  test_LOC_f1               =     0.7012
  test_LOC_number           =        614
  test_LOC_precision        =     0.7088
  test_LOC_recall           =     0.6938
  test_ORG_f1               =     0.6392
  test_ORG_number           =        525
  test_ORG_precision        =     0.6155
  test_ORG_recall           =     0.6648
  test_PER_f1               =     0.7462
  test_PER_number           =        790
  test_PER_precision        =     0.7443
  test_PER_recall           =     0.7481
  test_loss                 =     0.2144
  test_overall_accuracy     =     0.9343
  test_overall_f1           =     0.7021
  test_overall_precision    =     0.6962
  test_overall_recall       =     0.7081
  test_runtime              = 0:00:08.44
  test_samples_per_second   =    102.675
  test_steps_per_second     =     12.908

Macro f1 score:: 0.6955469427691648
```

IndicNER During Training

```
train_result = trainer.train()
metrics = train_result.metrics
```

[3000/3000 46:20, Epoch 3/3]

Step	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
500	0.723500	0.190422	0.802007	0.829335	0.815442	10213	0.673067	0.662681	0.667834	9786	0.768656	0.837245	0.801486	10568	0.750718	0.778716	0.764460	0.945071
1000	0.161500	0.178656	0.811047	0.846862	0.828567	10213	0.674806	0.694666	0.684592	9786	0.804291	0.837245	0.820437	10568	0.765462	0.794811	0.779861	0.947673
1500	0.141300	0.178792	0.822199	0.850778	0.836245	10213	0.683327	0.690170	0.686731	9786	0.802456	0.841030	0.821290	10568	0.771727	0.795989	0.783670	0.948147
2000	0.133700	0.176748	0.818343	0.857045	0.837247	10213	0.679073	0.697323	0.688077	9786	0.801317	0.840367	0.820378	10568	0.768443	0.800144	0.783973	0.948331
2500	0.123500	0.180346	0.810089	0.861647	0.835073	10213	0.672086	0.697016	0.684324	9786	0.802645	0.838569	0.820214	10568	0.763829	0.800962	0.781955	0.947719
3000	0.121200	0.179577	0.814822	0.859101	0.836376	10213	0.678874	0.697118	0.687875	9786	0.803396	0.837150	0.819926	10568	0.767947	0.799653	0.783480	0.948143

IndicNER During Evaluation

```
***** eval metrics *****
  epoch                   =        3.0
  eval_LOC_f1             =     0.8364
  eval_LOC_number         =      10213
  eval_LOC_precision      =     0.8148
  eval_LOC_recall         =     0.8591
  eval_ORG_f1             =     0.6879
  eval_ORG_number         =       9786
  eval_ORG_precision      =     0.6789
  eval_ORG_recall         =     0.6971
  eval_PER_f1             =     0.8199
  eval_PER_number         =      10568
  eval_PER_precision      =     0.8034
  eval_PER_recall         =     0.8371
  eval_loss               =     0.1796
  eval_overall_accuracy   =     0.9481
  eval_overall_f1         =     0.7835
  eval_overall_precision  =     0.7679
  eval_overall_recall     =     0.7997
  eval_runtime            = 0:02:41.01
  eval_samples_per_second =     83.596
  eval_steps_per_second   =       4.18
```

IndicNER on Train Dataset

```
***** train metrics *****
  test_LOC_f1             =     0.8751
  test_LOC_number         =      14841
  test_LOC_precision      =     0.8558
  test_LOC_recall         =     0.8954
  test_ORG_f1             =      0.772
  test_ORG_number         =      14082
  test_ORG_precision      =     0.7626
  test_ORG_recall         =     0.7816
  test_PER_f1             =     0.8739
  test_PER_number         =      15614
  test_PER_precision      =     0.8624
  test_PER_recall         =     0.8857
  test_loss               =     0.1055
  test_overall_accuracy   =     0.9667
  test_overall_f1         =     0.8422
  test_overall_precision  =     0.8289
  test_overall_recall     =      0.856
  test_runtime            = 0:03:57.54
  test_samples_per_second =     84.195
  test_steps_per_second   =       4.21

Macro f1 score:: 0.8403391476998284
```

IndicNER on Test Dataset

```
***** Test metrics *****
  test_LOC_f1               =     0.8136
  test_LOC_number           =        614
  test_LOC_precision        =     0.8097
  test_LOC_recall           =     0.8176
  test_ORG_f1               =     0.7134
  test_ORG_number           =        525
  test_ORG_precision        =      0.654
  test_ORG_recall           =     0.7848
  test_PER_f1               =     0.8745
  test_PER_number           =        790
  test_PER_precision        =     0.8427
  test_PER_recall           =     0.9089
  test_loss                 =     0.1444
  test_overall_accuracy     =     0.9567
  test_overall_f1           =     0.8097
  test_overall_precision    =     0.7764
  test_overall_recall       =      0.846
  test_runtime              = 0:00:10.25
  test_samples_per_second   =     84.584
  test_steps_per_second     =      4.293

 Macro f1 score:: 0.8005258053106568
```

ChatGPT Vs Ground Truth :

```
------------------------------------------------------------
----------
-----------------------------------ChatGPT-----------------
---------
------------------------------------------------------------
----------
```

	Label	Precision	Recall	F1-Score
0	O	0.917667	0.992579	0.953654
1	B-PER	1.000000	0.500000	0.666667
2	I-PER	0.750000	1.000000	0.857143
3	B-ORG	0.400000	0.400000	0.400000
4	I-ORG	0.000000	0.000000	0.000000
5	B-LOC	0.428571	1.000000	0.600000
6	I-LOC	1.000000	1.000000	1.000000
7	B-MISC	0.250000	0.034483	0.060606
8	I-MISC	1.000000	0.076923	0.142857

Macro-F1-Score : 0.520102990691226

IndicNer Vs Ground Truth

```
--------------------------------------------------------------------------
----------
-----------------------------------IndicNER-------------------------
----------
--------------------------------------------------------------------------
----------
    Label  Precision    Recall  F1-Score
0       O   0.912892  0.972171  0.941599
1   B-PER   0.615385  0.500000  0.551724
2   I-PER   0.500000  0.666667  0.571429
3   B-ORG   0.500000  0.600000  0.545455
4   I-ORG   0.625000  0.833333  0.714286
5   B-LOC   0.500000  1.000000  0.666667
6   I-LOC   0.333333  1.000000  0.500000
7  B-MISC   0.000000  0.000000  0.000000
8  I-MISC   0.000000  0.000000  0.000000
Macro-F1-Score :  0.49901765744316173
```

IndicBert Vs Ground Truth

```
    Label  Precision    Recall  F1-Score
0       O   0.906412  0.970315  0.937276
1   B-PER   0.400000  0.375000  0.387097
2   I-PER   0.714286  0.833333  0.769231
3   B-ORG   0.571429  0.800000  0.666667
4   I-ORG   0.750000  0.500000  0.600000
5   B-LOC   0.285714  0.666667  0.400000
6   I-LOC   1.000000  1.000000  1.000000
7  B-MISC   0.000000  0.000000  0.000000
8  I-MISC   0.000000  0.000000  0.000000
Macro-F1-Score :  0.5289189106393407
```

2. Hyper Parameter Tuning - INDICBert : (Check notebooks available in directory IndicBert FineTuning)

a. **Batch Size :** The batch size per GPU/XPU/TPU/MPS/NPU core/CPU for training.

Batch Size	Train Accuracy	Train Macro F1	Test Accuracy	Test Macro F1
8	0.9172	0.6589	0.9296	0.6856
10	0.9174	0.6573	0.9292	0.6822
16	0.9129	0.6326	0.9253	0.6674

Verdict : I choose 8 batch size because it is showing good results on test data.

b. **Learning Rate :** The initial learning rate for the optimizer.

Learning Rate	Train Accuracy	Train Macro F1	Test Accuracy	Test Macro F1
1e-5	0.902870	0.588049	0.9122	0.6031
3e-5	0.919003	0.660689	0.9331	0.6925
5e-5	0.922825	0.680018	0.9332	0.7006

Verdict : I choose 5e-5 learning rate because it is showing good results on train and testdata.

c. **Weight Decay :** Weight decay is a regularization technique used in training neural networks, which penalizes large weights by adding a decay term to the loss function.

Weight Decay	Train Accuracy	Train Macro F1	Test Accuracy	Test Macro F1
0.01	0.924143	0.683648	0.9364	0.7167
0.001	0.920593	0.667584	0.9331	0.7013

Verdict : I choose 0.01 weight decay because it is showing good results on train and test data.

3. Hyper Parameter Tuning - IndicNER((Check notebooks available in directory IndicNer FineTuning))

a. **Batch Size :** The batch size per GPU/XPU/TPU/MPS/NPU core/CPU for training.

Batch Size	Train Accuracy	Train Macro F1	Test Accuracy	Test Macro F1
8	0.944745	0.770868	0.9505	0.7842
10	0.945593	0.773796	0.9526	0.7922
16	0.947775	0.769607	0.9501	0.7879

Verdict : I choose 10 batch size because it is showing good results on test data.

b. **Learning Rate :** The initial learning rate for the optimizer.

Learning Rate	Train Accuracy	Train Macro F1	Test Accuracy	Test Macro F1
1e-3	0.82	0.0	0.83	0.0
1e-5	0.948061	0.782507	0.9578	0.8131
3e-5	0.947703	0.783706	0.9554	0.8064
5e-5	0.946454	0.779012	0.9548	0.8092

Verdict : I choose 1e-5 learning rate because it is showing good results on train and testdata.

c. **Weight Decay :** Weight decay is a regularization technique used in training neural networks, which penalizes large weights by adding a decay term to the loss function.

Weight Decay	Train Accuracy	Train Macro F1	Test Accuracy	Test Macro F1
0.01	0.947374	0.781102	0.9549	0.8034
0.001	0.948061	0.782507	0.9578	0.8131

Verdict : I choose 0.001 weight decay because it is showing good results on train and testdata.

4. Learning from this comparison:

a. Training Vs Evaluation : For both models IndicBert and IndicNER both models are not able to deliver accuracy and results at test time.
b. IndicNer Performs better on NER Task but still not able to deliver when compared with ground truth.
c. ChatGPT performs better than IndicBERT and IndicNER on Ground Truth because IndicBERT and IndicNER is trained on very small corpus compared to ChatGPT.
d. When running for Five Epochs , we can see training loss decreases but validation loss increases which maybe due to “Overfitting”.

Epoch	Training Loss	Validation Loss
1	0.214300	0.222826
2	0.197200	0.205257
3	0.178200	0.201126
4	0.160700	0.201779
5	0.144300	0.205040

Some of Reasons for Poor Results by IndicNER , IndicBert :-

- **Contextual Ambiguity** : Our model may not work for ambiguous cases such as ‘Ram’ , it may refer to the name of a person or the Hindu god also ‘Pushpa’ can be proper nouns or a flower so it depends on context.
- **Lack of Data** : may overfit the data.
- **No Capitalization** : Unlike English , Hindi does not have capitalization to help assist in NER tasks
- **Spelling Variations** : Spelling of words may depend on locality for example : vanaspati or banaspati both are used for plants.

- **Free Word Order:** Languages like Hindi, which follow a free word order, make the NER task more challenging as computational approaches can not be complemented with a pattern of PoS tags, or strict word order.