Towards Robust Named Entity Recognition for Bangla Language

HAZ Sameen Shahgir[1] **Ramisa Alam**[1] **Md. Zarif Ul Alam**[1]

[1] Department of CSE, BUET

Abstract

In this competition, we present two different approaches to tackle the Named Entity Recognition (NER) task on a Bangla dataset. The dataset was in the .conll format and consisted of 15300 sentences for training and 800 sentences for validation.

We performed an Exploratory Data Analysis (EDA) on the dataset, which revealed that the dataset had 7 different NER tags, with notable presence of English words, suggesting that the dataset is synthetic and likely a product of translation.

To tackle the NER task, we used the Conditional Random Fields (CRF) algorithm, which is commonly used for sequential labeling tasks. We performed feature engineering, which is an important step in building machine learning models, particularly for natural language processing tasks such as named entity recognition. We experimented with a variety of feature combinations including Part of Speech (POS) tags, word suffixes, Gazetteers, and cluster information from embeddings.

However all linguistic patterns are not immediately apparent or even intuitive to human and this is why Deep Learning has proved to be the more effective model in NLP, including NER. Deep learning models can automatically learn features from the input data, reducing the need for manual feature engineering. We used the BERT/ELECTRA model BanglaBERT (large) fine-tuned for NER, which achieved F1 Score 0.79 on the validation set.

Data Analysis

The provided dataset was a labeled Bangla dataset in the ċonll format where each word had a corresponding NER tag and sentences were separated with empty lines.

The train set consists of 15300 sentences and the validation set has 800 sentences. The length of the sentences in both sets varies from 2 words to 35 words with the average length being 12 words.

There are 7 different NER tags in the given dataset with LOC (3804), GRP (6653), PROD (5152), CW (5001), CORP (5299), PER(6738), and O (170K)

There were some English words in the English alphabet and many more English words transliterated into Bangla, suggestion that the dataset is synthetic and likely a product of translation.

Details EDA can be found in the EDA notebook of the GitHub repository.

Feature Based Learning

Before the advent of deep learning, feature engineering was a crucial step in building machine learning models, particularly for natural language processing tasks such as named entity recognition. Traditional machine learning algorithms rely on a fixed set of input features to make predictions. These algorithms cannot automatically learn useful representations from raw data, as deep learning models can.

Conditional Random Fields

Conditional Random Fields (CRF) is a type of machine learning algorithm that is commonly used for sequential labeling tasks, such as named entity recognition, where the goal is to assign a label to each word in a sentence. (Ekbal, Haque, and Bandyopadhyay 2008) CRF works by modeling the probability of a label sequence given the input feature sequence.

In CRF, feature engineering is important because it directly affects the quality of the input features, which in turn affects the performance of the model. Features that are relevant to the task and provide useful information about the input data will lead to better performance of the CRF model.

Let $x = x_1, x_2, ..., x_n$ be the input feature sequence, where x_i is the feature vector of the i-th word. Let $y = y_1, y_2, ..., y_n$ be the corresponding label sequence, where y_i is the label of the i-th word. Let $\Phi(x, y)$ be a feature function that maps the input feature sequence and the label sequence to a real-valued feature vector. Let w be the weight vector that parameterizes the feature function.

The goal of CRF is to find the most likely label sequence given the input feature sequence, which can be formulated as:

$$\arg\max_y P(y|x) = \arg\max_y \left(\frac{exp(w^T \Phi(x, y))}{\sum_{y'} exp(w^T \Phi(x, y'))} \right)$$

Feature Engineering

We developed our system by experimenting with a variety of feature combinations. The goal of selecting these features was to improve the ability of our classifier to generalize to new data and handle large amounts of data.

It is important to note that choosing features for Bangla Language turned out to be somewhat challenging compared to English. This is because some of the features that work well for English can't be used for Bangla. For example, it makes sense to check if a word is a title because it could be Person (PER), Location (LOC) etc. But, there is no such concept in Bangla. So, we had to choose the features carefully.

The details of the collection of features used for the NER task are provided below.

- **Part of Speech (POS)**: POS tags can provide useful information about the grammatical structure of the sentence and the role of a word in it. POS tags can be used as a feature in CRF to help disambiguate named entities from other types of words. Here we have used the Bengali POS Tag model, which is a CRF based POS tagger. The POS tagger was trained with nltr dataset with 80% accuracy. For each word, we have used the POS tag information for that word and k words before and after it. We have experimented with several values of k, and we concluded that $k > 3$ seems to overfit. So, we used $k = 2$ in the final model for generating features.
- **Word suffix**: Word Suffix is an important feature in case of Bangla NER. For example, locations (LOC) can end with suffixes আলয়, পুর, বাড়ি, লিয়া, পারা. Similarly, most female names in Bangla end with আ. We leverage the suffix information for this kind of case. And our experiment shows that the macro F1 score increases by a significant margin in our validation set.
- **Word prefix**: Word Prefix is similarly important information for NER. We used a fixed-length prefix of the current and/or the surrounding words.
- **k neighbour words**: Neighbouring words can carry useful information about NER of a word. We added the previous and next k words of the current word as a feature for different values of k. Based on the validation results, we used $k = 2$ in the final model for generating features.
- **Cluster Information from Embeddings**: Instead of using Word Embeddings directly, we took a slightly different approach. As the dataset is diverse and there are domain shifts, as well as the out-of-vocabulary words, we didn't think it would be a good approach to use them directly. And if we were to do that, the model won't be robust to unseen data. Our intuition for this feature was that when we meet unknown data, it will be closer to cluster where another NER of its kind belongs.

  So, we went for the Word Clustering mishra2016semi approach. Our algorithm is as follows -

  1. Collect and preprocess a large dataset of Bangla text for training the NER model.
  2. Train a word embedding model on the dataset, such as word2vec, GloVe, or fastText. The embeddings will capture the semantic and syntactic information of the words in the text.
  3. Use the trained embedding model to generate embeddings for each word in the dataset.
  4. Apply clustering techniques, such as k-means or hierarchical clustering, on the word embeddings to group similar words together.
  5. Train the CRF NER model on the dataset, using the word embeddings and cluster information as features.

  Here, we have used word2vec for extracting the word vectors and used k-means for clustering. We have also experimented with soft clustering using Gaussian Mixture Models (GMM), but because of the runtime overhead, we discarded that.
- **Gazetteer Lists using Bangla t5 NMT**: We used the idea of Gazetteer Lists to further improve our model. The problem was that there were no Gazetteer lists for Bangla to get started. We could have created that with a semi-supervised approach, but considering the domain and diversity of our dataset we took a different approach. We used the Bangla t5 NMT model (Hasan et al. 2020) developed by CSE BUET to create a new set of Bangla Gazetteers. The reasoning was that our dataset had lots of words like নিউ ইয়র্ক, চেকোস্লোভাকিয়া, অ্যারিজোনা. If we were to curate Bangla Gazetteers only using Bangla data, we would not be able to get this kind of data. The CSE BUET Bangla t5 model which was fine-tuned for Neural Machine Translation (NMT) task excels at this kind of task. When given names of places and creative words it is able to generate proper transliteration instead of just translating it to a Bangla word that doesn't make sense. We had score improvements in the creative word category using this method.
- **Digit** We also tried using digit information as a feature. But from our experiment, we saw that the impact was not significant.

Deep Learning Model

For the deep learning model, the model we chose was BanglaBERT (large), a BERT/ELECTRA model with 330M parameters pretrained on Bangla2B+ pretraining corpus. This model performed comparably to XLM-R (large) in NER (XLM-R (large) F1 score 78.39 vs. BanglaBERT (large) F1 score 79.20 on Semeval NER Bangla Dataset) while having fewer parameters (330M vs. 550M).

Furthermore, ELECTRA(Clark et al. 2020) is based using a generator-discriminator paradigm which makes

it uniquely suitable for low compute environments. "Masked language modeling (MLM) methods, such as BERT, involve corrupting input by replacing some tokens with [MASK] and training a model to reconstruct the original tokens. These methods have been shown to produce good results when transferred to downstream NLP tasks, but they typically require large amounts of computational resources. An alternative approach, called replaced token detection, is proposed as a more sample-efficient method of pretraining. Instead of masking input, this method corrupts it by replacing some tokens with plausible alternatives sampled from a small generator network. A discriminative model is then trained to predict whether each token in the corrupted input was replaced by a generator sample or not. Experimentation demonstrates that this new pretraining task is more efficient than MLM because the task is defined over all input tokens rather than just the small subset that was masked out. This results in contextual representations that substantially outperform those learned by BERT, given the same model size, data, and computational resources. This advantage is particularly pronounced for small models. For example, the authors (Clark et al. 2020) showed that a model trained on one GPU for 4 days outperforms GPT, which was trained using 30x more compute, on the GLUE natural language understanding benchmark. The approach also works well at scale and performs comparably to RoBERTa and XLNet while using less than 1/4 of their compute and outperforms them when using the same amount of compute.

We started our preliminary tests on BanglaBERT (base) and BanglaBERT (large), tuning various hyperparameters. BanglaBERT (large) consistently outperformed its smaller counterpart, given enough training cycles to reach convergence. As such, BanglaBERT (large) was the final choice.

Results

Feature Based Learning

We iteratively added features that improved our score. The results can be found in Table 1.

Deep Learning Model

Several strategies were explored to improve the robustness of our model. We first began by tuning maximum sequence lengths and batch sizes. The longest sentence in the training set was 34 words and allowing a maximum sequence length of 64 allows for dataset to fit into the model without truncation.

Analysis

Feature Based Learning

The problem with Feature Based Learning Models is that it is hard to scale and generalize for unseen datasets. Fetaures like Gazetteer Lists and Word Embedding based clusterings help generalize the model. We saw that feature based models struggles to identify CW tags. This is intuitive because these words depend on the context. And the handcrafted features are not able to identify that.

Deep Learning Model

Custom CrossEntropyLoss Weights Due the the data imbalance in the dataset, we tried a weighted CrossEntropyLoss function using the definition.

$$w_i = 1 - \frac{n_i}{\sum_{j=1}^{N} n_j}$$

However, after training for 4 epochs, the F1 score was 0.74 which is lower than the unweighted model (0.78) while all other hyper-parameters remained the same. It it possible a different weight function could perform better, which is left as future work.

Oversampling Certain Tags On un-augmented data, our selected model had the worse F1 scores on "CW" and "PROD" tags. We tried an oversampling policy of sampling sentences with "CW" and "PROD" two times. This resulted in no noticeable improvement, rather increased training time.

Knowledge Base To overcome the problem mentioned for Creative Word (CW) tags, we wanted to leverage external knowledge. To do that we created a knowledge base from Wikipedia using Bangla Wikimedia dump. But the problem was that if we do knowledge retrieval and add context paragraphs with the sentence, the dataset becomes 6x large. And the max_sequence_length needs to be large too. Given our aws memory limit it is not possible to train the model with the aforementioned constraints.

References

Clark, K.; Luong, M.-T.; Le, Q. V.; and Manning, C. D. 2020. Electra: Pre-training text encoders as discriminators rather than generators. *arXiv preprint arXiv:2003.10555*.

Ekbal, A.; Haque, R.; and Bandyopadhyay, S. 2008. Named entity recognition in Bengali: A conditional random field approach. In *Proceedings of the Third International Joint Conference on Natural Language Processing: Volume-II*.

Hasan, T.; Bhattacharjee, A.; Samin, K.; Hasan, M.; Basak, M.; Rahman, M. S.; and Shahriyar, R. 2020. Not Low-Resource Anymore: Aligner Ensembling, Batch Filtering, and New Datasets for Bengali-English Machine Translation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2612–2623. Online: Association for Computational Linguistics.

Feature	F1 Score
POS Tagger, Suffix	0.56
POS Tagger, Suffix, k-Neighbor Words	0.62
POS Tagger, Suffix, k-Neighbor Words, Gazetteer Lists	0.689
POS Tagger, Prefix, Suffix, k-Neighbor Words	0.692
POS Tagger, Prefix, Suffix, k-Neighbor Words, k-means clustering	0.72

Table 1: Iteratrive Improvement of Feature Based Learning Model

Model	Batch Size	Max Seq Length	Epoch	F1 Score on dev.txt
base	16	128	3	0.73
large	16	128	3	0.77
large	32	64	3	0.76
large	16	128	6	0.78
large	32	64	6	0.79
oversampled+large	16	128	6	0.78
SemEval2023data+large	32	64	4	0.78
SemEval2023data+weights+large	32	64	4	0.74
SemEval2023data+large	**32**	**64**	**6**	**0.79**

Table 2: F1 Scores of different DL models

Metric	Value
P-GRP	0.8252427184466011
R-GRP	0.7203389830508469
F1-GRP	0.7692307692307188
P-CORP	0.7999999999999994
R-CORP	0.787401574803149
F1-CORP	0.793650793650743
P-CW	0.7563025210084028
R-CW	0.7499999999999993
F1-CW	0.7531380753137571
P-PROD	0.6783919597989946
R-PROD	0.7105263157894732
F1-PROD	0.6940874035989213
P-LOC	0.7909090909090902
R-LOC	0.8613861386138605
F1-LOC	0.8246445497629825
P-PER	0.9078947368421045
R-PER	0.9583333333333326
F1-PER	0.9324324324323818
Precision	0.7858910891089108
Recall	0.7937499999999998
F1	0.7898009950248256
MD-R	0.8775
MD-P	0.8688118811881188
MD-F1	0.8731343283581589
ALLTRUE	800
ALLRECALLED	702
ALLPRED	808

Table 3: Performance of Final Model

Development Log

BUET Semantic Shenanigans

20/01/2023 - 11 am - Zarif

- Wkiiann dataset - https://www.tensorflow.org/datasets/catalog/wikiann

- HuggingFace available models
    - https://huggingface.co/sagorsarker/mbert-bengali-ner
    - https://huggingface.co/Suchandra/bengali_language_NER
        - Fine Tuning Code available in github: https://github.com/SuchandraDatta/bengali_language_NER/blob/main/ner-bengali.ipynb
- Another BD NER Dataset - https://github.com/MISabic/NER-Bangla-Dataset
- A recent **survey** paper on BD NER - Bengali Named Entity Recognition: A survey with deep learning benchmark
    - Dataset From this paper - https://github.com/Rifat1493/Bengali-NER
    - ^^ **this paper actually has references to work that was done by feature engineering. Need to read those.**

20.01.2023 11 am - Ramisa

- Fine tune HuggingFace models for NER
    - mt5: https://tsmatz.wordpress.com/2022/10/24/huggingface-japanese-ner-named-entity-recognition/
    - XLMRoberta: https://tsmatz.wordpress.com/2022/10/24/huggingface-japanese-ner-named-entity-recognition/

  20 epochs : val F1 = 43.35

20.01.2023 - 1:00 pm - Sameen

Trying to adapt BanglaBERT for NER. Converted data to tokens, ner_tags .csv csebuetnlp/banglabert normalizer removes double space which disalignsthe words and tags. Fixed this manually for 21 samples in train set.

20.01.2023 - 2:00 pm - Sameen

Custom function to deal with tokenizer splitting words into many. Redundant.

Found csebuetnlp/banglabert token_classification code

Converted data into a suitable json format for that.

banglabert tokenizer splits words like “(word)” = “(” , “word”

20.01.2023 - 4:00 pm - Sameen

BanglaBERT base: batch_size 16 max_length 128 epoch 3 0.73

BanglaBERT large: batch_size 16 max_length 128 epoch 3 0.77

BanglaBERT large: batch_size 32 max_length 64 epoch 3 0.76

Runtime: 43 minutes vs. 65 minutes

20/01/2023 - 7:00 pm - Zarif

- http://noisy-text.github.io/2017/files/
- Did literature review on NER Feature Engineering
- Found out about CRF
- Did CRF with only a few handcrafted features
- Got top F1 about 0.6

20/01/2023 - 8.00 pm - Ramisa

SemEval 2022 Task 11

- https://assets.amazon.science/fb/3f/7b2f48744d428de6f442e4477524/semeval-2022-task-11-multilingual-complex-named-entity-recognition-multiconer.pdf
- Best 2:
    - (USTC-NELP): https://arxiv.org/abs/2203.03216
    - (DAMO NLP): https://arxiv.org/abs/2203.00545

- Insights:
    - Add external knowledge for CW, PROD tags
    - Use Gazetteer

20/01/2023 - 10.00 pm - Sameen
Oversampling Attempt on CW and PROD tags:

batch_size 16 max_length 128 epoch 6 0.78

21/01/2023 - 12:00 am - Ramisa

Useful features for feature-based model from:
https://www.researchgate.net/publication/228576012_Named_Entity_Recognition_in_Bengali_A_Multi-Engine_Approach

Added features:

- Previous 2 words, Next 2 words
- Suffixes and prefixes of current word
- Suffixes and prefixes of next words
- Suffixes and prefixes of prev words
- Current word length
- Current word index
- Current word frequency in train set
- POS tag of current, prev, next words
- Beginning of sentence tag
- End of sentence tag

Tried:

- Adding embeddings
    - word2vec, glove for bn - error
    - Fasttext - taking too long

2201/2023 - 3:00 am - Sameen

Inherited the huggingface Trainer function and added a class weight to the loss function.

23/01/2023 - 3:00 am - Sameen

SemEval2023data+weights+large batch size 32 max length 64 epoch 4 0.74
Score lower than unweighted loss function