A report on Named Entity Recognition

NLP Mavericks

Jayanta Sadhu
Ayan Antik Khan
Sadat Shahriyar

Data Analysis

For any Named Entity Recognition (NER) task, data analysis is a crucial subtask. In our dataset, analysis revealed a number of interesting features that motivated us to make decisions in order to get better metrics after training our model.

	tags	counts
0	B-CORP	2597
1	B-CW	2156
2	B-GRP	2405
3	B-LOC	2351
4	B-PER	2606
5	B-PROD	3188
6	I-CORP	2701
7	I-CW	2843
8	I-GRP	4248
9	I-LOC	1453
10	I-PER	3132
11	I-PROD	1964
12	O	160236

Insights

We found that the data had much more tokens of tag **O** than the rest. So naturally the data is skewed towards the **O** class.

We also noticed that there are no null values. There were some garbage values like hyphens(-) which we cleaned while preprocessing.

METHODOLOGY

Preprocessing

For our Deep Learning Model (**banglabert-large),** we did a number of preprocessing steps before sending our data to the model.

1. We normalized our dataset using a normalizer for bangla text.
    a. In this step, we cleaned the **punctuations** from the data.
2. We tried to perform **stemming and lemmatization** on the data tokens using regex and python modules but in this process too much contextual information was being lost.
3. We made a positional analysis on the tokens' and tried to clean the dataset by removing tokens that had difference of tags from a certain position up to a certain margin, but that didn't give any good result.
4. We took a library of bangla **stopwords** from the internet and removed the stopwords to clean the dataset hoping that it would improve the performance. But it took away too much data from the dataset to work on it.
5. Below are some approaches that we tried on our quest to get a hold of the dataset:
    a. Bootstrapping: We tried to run bootstrap sampling on the dataset. But decided against it as the data is heavily skewed towards 'O' class, which prevented us from getting any balanced dataset.
    b. One vs All training Method: In this approach we tried to keep a minority class in one group and the rest into another group. The plan was to run different models for different classes and apply ensembles. But the method would make a highly imbalanced dataset of ratio around 2:98 (some class vs other).
6. Finally we tried augmenting the data to replicate the distribution of the dataset and increase the samples for non-other classes. The augmentation methods that we used are as follows:

a. **Swapping Words**: We randomly chose an index inside the tags list and swapped it with the next word in the right cyclic manner upon some considerations. We made sure that the two corresponding words are not of different groups and corrected [B and I] tags and [I and O] for swapping tokens of the same category
b. **Label-Wise Token Replacement:** We made a dict of category class and populated it with tokens from that respected category. We choose token index at random and replaced it with the token from the respective category and took multiple instances of this operation on the same data instance
c. **Synonym Replacement:** We used the **banglanltk** library to generate synonyms for a token in the dataset and used it for augmenting the data
d. **Random Insertion:** We choose random indices inside a list and replace that with a token from a random class category. We made the following checks after all token insertions:
    - Whether a different category I follows a B tag. If so, change the later I to B of that respective class.
    - Whether a B follows a B from the same category. In that case, modify the latter B into an I.
    - If an I is followed by an O, change that I into a B
    - If a B follows an I with the same group, the next B is converted to an I.
    - If an I follows an I of different category, then the later I is modified to be a B

    We do not insert any other type object because there is 'O' class in abundance

e. **Random Deletion:** We select random positions of the token list and remove that from the data with a certain probability.

Deep Learning Based Model Training

We used pretrained BanglaBert large model from hugging face for our training. We used some hyperparameter tuning in our training process. We saved the checkpoints locally and measured the best model using **Macro_f1**. We used the predefined trainer script on the official github repo of buetcsenlp on the task of token classification

Machine Learning Based Model Training

For the machine learning based model, we tried to analyze the data using df.

Firstly we tried to ensemble different models with our preliminary data, those include Stochastic Gradient Descent(SGD), Naive Bias Classifier, Passive Aggressive Classifier, SVM and XG-Boost. But due to the small size of our data this didn't work.

Then we tried to perform some feature engineering on the data. We followed feature extraction methodology from a paper mentioned in the resources section. We tried to perform CRF on the data. But that also didn't do much good.

Next we tried the model on our augmented data of size 80k. This produced a good result from the model. We fit multiple ML models with varied f1 scores, we noticed that the models performed much better when we were training the models with our augmented data.

RESULT

Deep Learning Based Black Box Approach

We got our best results after 3 Epochs.

Strength: Our model captures the **CW(Creative Words)** very well. We find the score

eval_CW_f1 = 0.6531

Weakness: We think we could not train our model enough. We were hopeful that we would have gotten better scores if we trained our final data augmentation for much longer. For which the time constraint was a big barrier.

Training result

'eval_macro_avg_f1': 0.7598513940656014

Validation result

predict_macro_avg_f1 = 0.7938

Machine Learning Based White box Approach

We got good results after training the SVM Classifier with 100,000+ tokens. We found the **macro_avg_f1_score : 0.68**

FUTURE WORK

Our main constraint was time. We did not have enough time to train our augmented data with the deep learning model. We were hopeful that we would get much better results if we could train for multiple epochs over several hours.

Training a huge augmented dataset (Keeping overfitting in mind) would hopefully increase the metric of our model.

REFERENCES

1. banglabert-large(CSE, BUET)
2. Text normalizer for Bangla
3. banglanltk
4. EDA and Text Preprocessing Bert
5. Semi-supervised Bootstrapping approach for Named Entity Recognition
6. Named Entity Recognition with Scikit Learn