Named Entity Recognition using Deep Learning Models and Feature-Based Models
Team: Linguistic Mavericks

MD Ajmain Mahtab Sheikh Ayatur Rahman Anika Tahsin Shemonty

January 21, 2023

Natural Language Processing Hackathon

Organized by Bangladesh Open Source Network (BdOSN)

1 Introduction

Named Entity Recognition is one of the most popular problems in natural language processing. The basic idea of the problem is to assign labels to tokens in a body of text. The challenge in this problem lies in the fact that the label of a token depends on its context and its relations with other words. This problem can be tackled with both deep learning based models as well feature-based models. In this report, we showcase two models that can perform Named Entity Recognition. The first one is the XLM-RoBERTa model fine-tuned on the dataset provided which achieved an F-1 score of 0.8. The second one

2 Exploratory Data Analysis

The training data is formatted so that on each line, we have a token followed by two dashes which in turn is followed by the label - all separated by a space. Each sentence was separated by an empty line. We found that there were 13 unique labels for the tokens. We noticed that the dataset was imbalanced with the vast majority of tokens being of the "O" class. The sentences varied in length as well. Our models had to be robust enough to not tackle these challenges.

3 Methodology

XLM-RoBERTa
We fine-tuned various pre-trained models including Bangla-BERT using a wide array of hyperparameters.
Our best performing model was the XLM-RoBERTa model which we fine-tuned on our dataset. We used the large version of XLM-RoBERTa which has well over 500 million parameters. The large size of the model and extensive pretraining of XLM-RoBERTa on various languages allows it to understand language and context on a much deeper level than many other pretrained models.

We preprocessed the data by tokenizer the sentences with the XLM-RoBERTa-Large tokenizer. Given that this was a subword tokenizer, we had to assign the correct labels to the subwords. Additionally, we added special tokens to the sentences so that the model could process our input. For the purpose of assigning the correct label to the subwords, we used word ids provided by the tokenzier to keep track of the origin of the subwords. We assigned the correct label only to the first subword of a given word. As for the special tokens and other subwords, we labelled them with -100 so that the model does not consider those tokens during the fine-tuning process. Finally, we padded all sentences to the length of the largest sentence in the dataset.

For the pre-training process, we used a learning rate of 2e-5, batch size of 16, and weight decay of 0.01. We trained the model for 10 epochs. After training, we achieved an F-1 score of 0.8 on the provided dev set.

Feature-based Models
For the featured-based model, we used Decision Tree Classifier, Random Forest Classifier, Gradient Boosting Classifier, XGBClassifier, Naive Bayes, SVC. The best model among these was Random Forest Classifier.
We used word embeddings based on a text corpus on our data and separated words and labels of each row. Then we found all the labels, and encoded them. After labeling we fit our data and ran through the models which gave a poor F-1 macro result. We also applied SMOTE and ADASYN, and ran the models again. But it did not help much with the score and had a poor result.
Since, word2vec approach was not giving a satisfactory result we switched to a different approach for data preprocessing. We separated the labels and encoded them by setting different values for each category. Then we pushed the data into the pandas dataframe and ran our model, Random Forest Classifier. After fitting the data we predicted on our labeled dev dataset. This gave us a F-1 macro score of 0.548040 on dev dataset which is way better than the word2vec approach which gave us a F-1 marco score of around 0.03.

4 Results

Model	F-1 Score
1) Fine-tuned RoBERTa-XLM	0.8
2) Feature-based RandomForest	0.55

We can observe that the fine-tuned RoBERTa-XLM is the best performing model. Due to extensive pre-training of the model on a large corpus of text, along with advanced training techniques, the model has a thorough understanding of language and context which allows it to successfully classify tokens.

5 Analysis

RoBERTa-XLM Since RoBERTa-XLM is based on the transformer architecture, its attention-based mechanism allowed the model to pay attention to parts of the sentences most important for classifying the token. Furthermore, since it was trained using masked language modelling, it can a much deeper idea about context that feature-based architectures simply do not have. Lastly, due the large number of parameters, it learning capacity is immense and as such was very successful in this task.
However, as mentioned before, the imbalanced nature of the dataset strongly inhibits the model's performance. Due to the large number of "O" labels in the dataset, it could not utilize its full learning potential during the fine-tuning stage.

As for feature-based model, Naive Bayes and logistic regression were too simple for the task, so did not perform well. XGBClassifier was used with and without hyper parameters tuning, and Gradient Boosting Classifier was used without hyper parameters tuning. XGBClassifier performed better than Gradient Boosting but could not outperform Decision Tree Classifier.
Since Decision Trees performed better, it indicated ensemble techniques that use decision trees would perform well. Random forest was marginally better while there was some improvement in performance after gradient boosting classifier was used.
Moreover, in both cases Random Forest Classifier was better than Gaussian Naive Bayes and Decision Tree classifier. Thus, a Random Forest Classifier was chosen.

Day 1:

[Ayat]

- preprocessing data for model pretraining
- will fine tune bangla bert
- figured out a way to address the issue of assigning labels to subwords

[Ajmain]

- Loaded the train, test data from ‘dev.txt’
- Inspecting the data
- Converted both the features and labels into numbers
- Ran a Random Forest classifier on the data
- Did the same preprocessing on it

[Anika]

- Preprocessed data with word2vec
- Ran SVC, Decision Tree Classifier
- F-1 macro score was very low so applied SMOTE and ADASYN on Decision Tree and compared differences
- Ran Gaussian Naive Bayes without hyper parameters and got F-1 score of 0.076

[Ayat]

- first round of training completed
- wrote a script for making inferences for the evaluation script - best f1 so far is 0.66

[Ajmain]

- Random Forest was too slow, so changed to Naive Bayes
- Precision, recall and f1_score was low so applied SMOTE

[Anika]

- Applied XGBClassifier without hyper tuning
- XGBClassifier with SMOTE and ADASYN
- Ran XGBClassifier with hyper parameter , got accuracy 0.371138 and F-1 score of 0.054593

Day 2:

[Anika]

- Tested out XGBClassifer with RandomSearchCV to find the best hyperparameter
- Applied SMOTE and ADASYN then compared with XGBClassifier report
- Switched from word2vec to converting feature and label into numbers
- Ran Decision Tree Classifier without hyper parameter tuning got accuracy of 0.886577 and F-1 score of 0.542764 on dev dataset

[Ajmain]

- Ran Random Forest Classifier
- Separated the ‘B’ and ‘I’ part from the labels, to halve the number of labels. Wrote codes to add the ‘B’s and ‘I’s.
- Created a way to store all the spaces and then put them back before giving the output
- Tried SMOTE, ADASYN and tried hyperparameter tuning Random Forest Classifier. But the model performed worse or marginally improved

[Ayat]

- tuning hyperparameters for model fine tuning
- switched from Bangla Bert and now trying out RoBERTa
- roberta boosted f1 to 0.71
- trained a model for 2 hours - got an f1 of 0.80
- handcrafted features for feature based model
- tried out features on various model - very poor performance on all of them