Named Entity Recognition

Project Report

Vishwajeet Mannepalli

Introduction

Named Entity Recognition(NER) is a subtask of NLP(Natural Language Processing) that mainly focusses on identifying and classifying entities in a given text, such as names of people ,organizations, locations, dates, or other specific information Which will be helpful for other ML problem like text summarization. For example., with the help of named entities, you can form different patterns which can help the model to understand more about the semantic of the text.

Why NER is important? In the field of Artificial Intelligence(AI), NER will let AI to understand the text even better by allowing them to prioritize relevant information and filter out irrelevant information. It also assists AI in understanding the context of a given text, which is vital for applications like machine translation, text classification, or recommendation systems. As matter of fact, In search engines, NER helps in providing more accurate and relevant search results by identifying entities in texts.

Why do you think the NER problem is challenging? Lot of named entities have multiple meanings and interpretations, making difficult for a model to correctly identify and classify them. There is also another challenging problem, with different kinds of words you get different kinds of tags. So, larger the dataset, more number of classes we need to predict. In this project we got 17 tags/ classes to predict. Hence, it is very difficult to get a good accuracy considering the fact that a model has to predict a word which could be any of the 17 classes we got.

The Dataset we have used in this project is already pre-processed and available in kaggle website. Dataset contains collection of random sentences.This dataset consists of four features which are, ‘Sentence #’, ‘Sentence’, ‘POS’, ‘Tag’. This dataset consists of 47,959 rows I.e., sentences and total number of words are roughly one million.

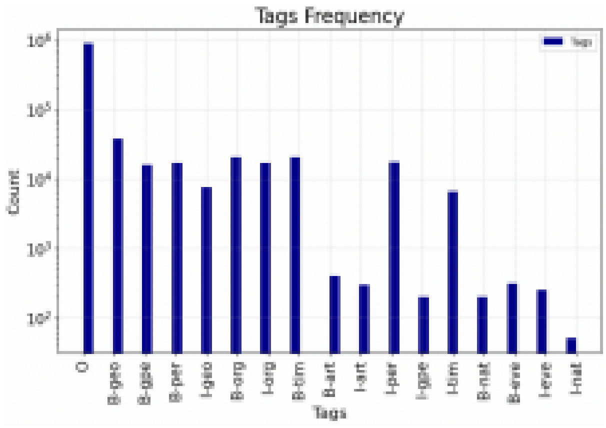


Fig 1 , Number of tags

Our dataset consists of 17 unique tags, which are ‘O’ is outside of a named entity, ‘B-geo’ is beginning of geographical entity, ‘B-gpe’ is beginning of a geopolitical entity (e.g., countries, cities),’B-per’ is beginning of a person’s name, ‘I-geo’ is inside a geographical entity, ‘B-org’ is beginning of an organization name, ‘I-org’ is inside of an organization name, ‘B-tim’ is beginning of a time-related entity (e.g., dates, days). ’B-art’ is beginning of an artifact name (e.g., books, movies), ‘I art’ is inside an artifact name, ‘I-per’ is inside a person’s name, ‘I-tim’ is inside a time-related entity, ‘B-nat’ is beginning of a natural phenomenon name (e.g., hurricanes), ‘B-eve’ is beginning of an event name (e.g., conferences, sports events), ‘I-eve’ is inside an event name, ‘I-nat’ is inside a natural phenomenon name. I have split the dataset into training, testing and validation datasets.

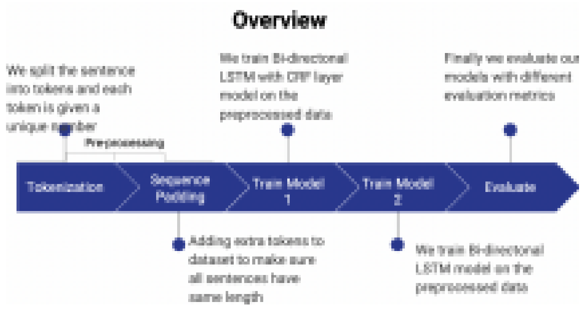


Fig 2, Overview of whole

The next pre-processing step we used in this project is sequence padding. Input data may consist of sequences of varying lengths, such as words, phrases, or sentences. Machine learning models, especially neural networks, often require fixed-length input vectors to perform optimally. Padding is typically done by adding a special "padding token" (usually represented by a value like 0) to the end of shorter sequences until they reach the desired length. Truncating involves

Solution used for this problem in this project are Deep neural networks. We used Bi-directional LSTM layers to build two models and we trained the dataset on them and finally models are used to predict random sentences.

First and Foremost, we looked into the dataset, it has four features of which we will be using only two which are, ‘Sentence’ and ‘Tag’. Sentence is collection of multiple words, whereas, ‘Tag’ is a collection of tags for a respective word in the sentence.

Pre-processing

As we are dealing with text data. We have used two preprocessing steps, which are tokenization and sequence padding.Tokenization is the process of breaking down a given text into smaller units called tokens. In natural language processing (NLP) and computational linguistics, tokens are usually words, phrases, or sentences that help machines understand and analyze human language. We have used word tokenization approach, Each and every token(word) has been assigned a unique number. So, now we have sentences as array of integers.

removing tokens from the end of longer sequences until they match the required length.

Bi-directional LSTM model

One of the models we have used here consists of Bi-directional LSTM layers. A bi-directional LSTM (Long Short-Term Memory) is a type of recurrent neural network (RNN) architecture that is designed to capture and learn patterns in sequential data more effectively than a traditional RNN. The neural network model used in this project has 2 layers of bi-directional and a single time distributed layer.

A TimeDistributed layer is used in conjunction with an LSTM layer (or other recurrent layers) when you want to apply a fully connected layer or a convolutional layer to every output time step of the LSTM independently. It essentially applies the same dense or convolutional layer to each time step of the LSTM output sequence, treating each time step as a separate input.

The parameters used in this model after fine tuning are Adam optimizer, sparse categorical cross entropy loss, accuracy metrics, 5 epochs, 64 batch size.

Bi-directional LSTM model with CRF layer

This is other model used in this project consist of every layer used in the first model but it additionally consists of CRF(conditional random field) layer as an output layer instead of time distributed layer. The key advantage of using a CRF layer as the output layer, as opposed to a simple dense layer followed by a 'softmax' activation function, is that CRFs can capture the dependencies between the output labels in the sequence. In many sequence labeling tasks, the label of a token depends not only on the input token but also on the surrounding labels. A CRF layer is designed to model these dependencies, enabling the model to make more informed predictions.

Results and observations

After training both models,I got the following observations,

- Both models gave me 98% accuracy but something was wrong with model with CRF layer.
- Model with CRF layer has been evaluated with other different evaluation measures like hamming loss, Jaccard Score and F1 score. By looking at fig 3 we can clearly tell that Majority class is begin predicting a lot than the minority classes.

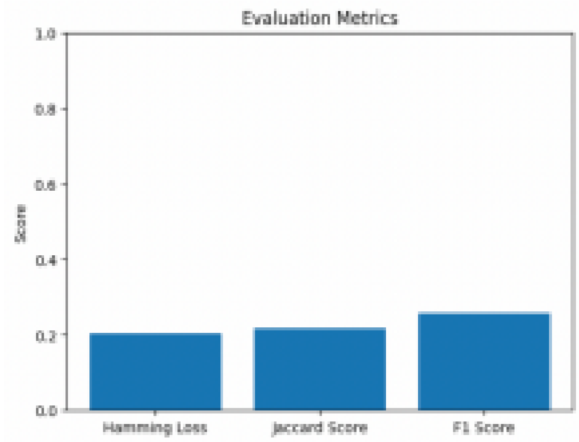
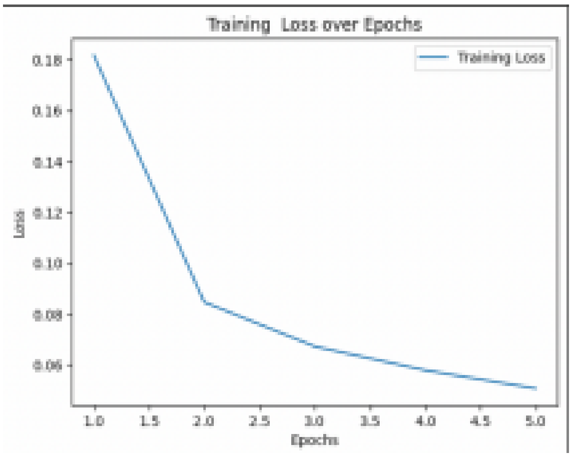


Fig 3

- Whereas, the Model with time distributed layer as output layer has been evaluated with different measures and got very good results. When you see the below fig we can clearly see that it has a good elbow curve for its loss function.

```
Precision: 0.98
Recall: 0.98
F1 score: 0.98
```

Jaccard score: 0.96



Model predictions on test cases

I have give 3 test cases for both model and here are the following results:

Model 1

Test case 1: Took sentence from the dataset

This is input sentence : Thousands of demonstrators have marched through London to protest the war in Iraq and demand the withdrawal of British troops from that country .

This is input sentence : Apple is looking to buy a London based startup for $1 Billion

This is the Predicted Tags : ['B-org', 'O', 'O', 'O', 'O', 'O', 'B-geo', 'O', 'O', 'O', 'O', 'O']

Test case 3:

This is input sentence : Apple London city

This is the Predicted Tags : ['B-org', 'B-geo', 'O']

Model 2(with CRF layer)

Test case 1: Took sentence from the dataset

This is input sentence : Thousands of demonstrators have marched through London to protest the war in Iraq and demand the withdrawal of British troops from that country .

This is the Actual Tags : ['O', 'O', 'O', 'O', 'O', 'O', 'B-geo', 'O', 'O', 'O', 'O', 'O', 'B-geo', 'O', 'O', 'O', 'O', 'O', 'B gpe', 'O', 'O', 'O', 'O', 'O']

This is the Actual Tags : ['O', 'O', 'O', 'O', 'O', 'O', 'B-geo', 'O', 'O', 'O', 'O', 'O', 'B-geo', 'O', 'O', 'O', 'O', 'O', 'B gpe', 'O', 'O', 'O', 'O', 'O']

This is the Predicted Tags : ['O', 'O', 'O', 'O', 'O', 'O', 'B-geo', 'O', 'O', 'O', 'O', 'O', 'B-geo', 'O', 'O', 'O', 'O', 'O', 'B gpe', 'O', 'O', 'O', 'O', 'O']

Test case 2:

This is the Predicted Tags :['O', 'O', 'O', 'O', 'O', 'O', 'B-geo', 'O', 'O', 'O', 'O', 'O', 'B-geo', 'O', 'O', 'O', 'O', 'O', 'B gpe', 'O', 'O', 'O', 'O', 'O']

Test case 2:

This is input sentence : Apple is looking to buy a London based startup for $1 Billion

This is the Predicted Tags : ['B-org',

'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O']

Test case 3:

This is input senctence : Apple London city
This is the Predicted Tags : ['B-org', 'B-geo', 'O']

Discussion

I have fine tuned the parameter for both models. I have tried to increase the accuracy of the model with changing number of epochs, number of base weights, activations functions(like replacing our tanh function with rely and leaky rely. I have also changed our loss function to SGD(stochastic gradient descent).
Well Bi-directional Neural networks w o r k e d w e l l o n n a m e d e n t i t y recognition. I would also try to implement BERT Transformer on this dataset in future. Transformer are the best model currently on sequence or text labeling .

Conclusion

From above results we can clearly tell the model without CRF layer works more accurately on this dataset.

References

https://www.kaggle.com/code/naseralqaydeh/named-entity recognition-ner-with-tensorflow/input

https://en.wikipedia.org/wiki/Bidirectional_recurrent_neural_networks

https://medium.com/mysuperai/what is-named-entity-recognition-ner-and how-can-i-use-it-2b68cf6f545d