Named Entity Recognition

Vishwashrisairam Venkadathiriappasamy
The University of Texas at Dallas
Richardson, Texas-75080, USA
vxv180043@utdallas.edu

Abstract

Named Entity Recognition (NER) refers to the task of extracting entities from text and tagging them with labels. This helps in adding semantic knowledge to the text, thereby helping to understand the subject of the text. This project aims to implement the task of extracting the named entities using a feature driven machine learning approach using Logistic Regression and using deep based learning model using LSTM.

1. Introduction

Named Entity Recognition is one of the most common problems in Natural Language Processing. There are large amount of unstructured text data available nowadays from traditional media sources as well as newer ones, like social media. These provide a rich source of information if the data can be structured. Named Entity Extraction forms a core subtask to build knowledge from semi-structured and unstructured text sources.
There are many more applications. NER can be used very effectively wherever information has to be selectively extracted from text. For example Named Entity Recognition can automatically scan the articles and reveal which major people, organization or place is being discussed. This is can be further used for automatically categorizing articles and smooth content discovery. One example for this is the customer feedback system where any complaint/feedback would be automatically routed to relevant branches based on the branch location mentioned in the feedback. If Named Entity Recognition can be run once on all the articles and the relevant entities (tags) associated with each of those articles are stored separately, this could speed up the search process considerably. So it can be used for designing efficient search algorithms. Another use case of Named Entity Recognition is the Recommendation system. For news publishers, using Named Entity Recognition to recommend similar articles is a proven approach. NER can be also be used for research paper segregation. Segregating the papers on the basis of the relevant entities it holds can save the trouble of going through the plethora of information on the subject matter.

There are few common methods that can be used for the entity extraction. Regular Expressions, Entity Lists and the Statistical Methods are among the few well known methods. Most of the NER systems are based on statistical models that incorporate machine learning techniques to classify the named entity. These include HMM based, CRF based, Logistic Regression and Neural Network based models and these have been found to be very good in doing the NER task. This project focuses on feature based machine learning approach using Logistic Regression and a deep learning based approach using LSTM network. The next section will describe the steps involved for this project.

2. Implementation

The goal of this project is to perform the task of NER tagging. That is if given a sentence, extract all relevant named entities and assign each named entity with relevant tag. For example given the below sentence:

"Jim bought 300 shares of Acme Corp. in 2006.", the NER would classify it as :

Jim -> Person; 300 -> Number; Acme Corp. -> Organization; 2006 -> Date

For this project we would be using CoNLL 2003 dataset. This dataset defines 4 tags: Person, Location, Organization and Miscellaneous. The dataset consists of sentences from Reuters' news stories/articles between August 1996 and August 1997.

Figure 1 gives a high level flow chart of the steps involved in the implementation.

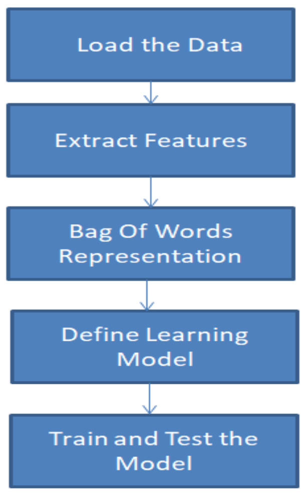
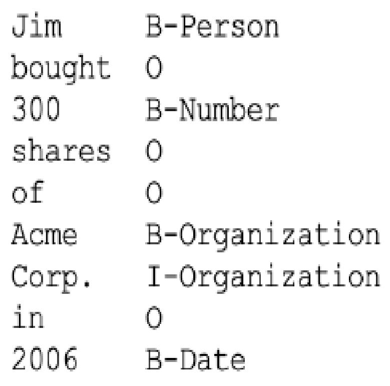


Figure 1 Flow chart of the tasks involved in the implementation

As seen from the figure, we can divide the process into five different steps: Loading the Data, Extracting/Finding Relevant features, Feature Representation using BOW model Building the machine learning models and finally training/testing the models.

2.1 Loading the Data

The first step involves loading the data as well as the labels from the dataset and creating a vocabulary that contains the integer encoding for a particular word. In the dataset, sentences are represented in the CoNLL form, where each new line indicates the beginning of a new sentence. Within each sentence, tokens are also new line-separated. Each token is associated with its gold NER tag. The gold NER tag is tagged using BIO format where:
I- prefix before a tag indicates that the tag is inside a chunk.
B-prefix before a tag indicates that the tag is the beginning of a chunk.
O tag indicates that a token belongs to no chunk



```
Jim     B-Person
bought  O
300     B-Number
shares  O
of      O
Acme    B-Organization
Corp.   I-Organization
in      O
2006    B-Date
```

Figure 2 Gold NER tagging format

Each line also contains some additional information like the gold POS tag and the syntactic head for each token.

2.2 Extracting the features

Once the data is loaded, for each tag we need to extract seveal features. We extract the following features for each tag in the dataset: the parts of speech tag, all the possible lemmas, hypernyms, hyponyms, holonyms, meronyms and antonyms of the tag. These features are extracted from the the wordnet which is available in the ‘nltk’ library.

Below is an example of how the feature would be represented for each tag:

[tag, [lemmas], [antonyms], [hypernyms], [hyponyms], [holonyms], [meronyms], [pos tags]]

where [] indicates a list

Simultaneously, we are also creating a dictionary that contains the encodings for all the words obtained so far. We could create a dictionary of 78368 unique words.

2.3 Bag of Words Representation

Once we extract the features for each tag, we use the bag of words model to represent the features. The previous section, gave an example of the sample feature representation of the extracted feature for a tag. Here, we merge all of these features into a single array of size equal to length of the unique words in the vocabulary + the number of unique POS tags in the dataset and encode it with count(numeric) values corresponding to the dictionary. The example below is a sample output of this step. Each tag would be represented by a vector of approximately 78400 features.

[1 0 1 1 0 0 . . . . 1 0 0]

2.4 Define the Learning Models

The next step is to define the model for classification. Here we are implementing two different learning models and compare their performance for the given dataset. The different models tried out in this project are:

1) Logistic Regression
2) RNN model

2.4.1 Logistic Regression

In natural language processing, logistic regression is the base-line supervised machine learning algorithm for classification, and also has a very close relationship with neural networks. The Logistic regression model implemented in this project uses log loss function and Stochastic Gradient Descent method as optimizer.

This model was run on the dataset containing 245821 tags in the dataset and for each tag is represented with approximately 78400 features. Because of the large number of tags and features, the program crashed because it ran out of memory. So for training these models we divided the input tags into batches of size 10000 and iteratively trained the model in batches. The SGDClassifeir available in sklearn library allows to incrementally train the logistic regression model.

2.4.2 LSTM RNN

Recurrent Neural Network (RNN) is a class of artificial neural networks where connections between nodes form a directed graph along a temporal sequence. This allows it to exhibit temporal dynamic behavior. RNNs can use their internal state (memory) to process sequences of inputs. This makes them applicable to tasks such as unsegmented, connected handwriting recognition or speech recognition. In the second part of the project we used a LSTM

RNN for learning the features. Figure 3 shows the architecture for the LSTM classifier. A common LSTM unit is composed of a cell, an input gate, an output gate and a forget gate. The cell remembers values over arbitrary time intervals and the three gates regulate the flow of information into and out of the cell. LSTM networks are well-suited to classifying, processing and making predictions based on time series data.

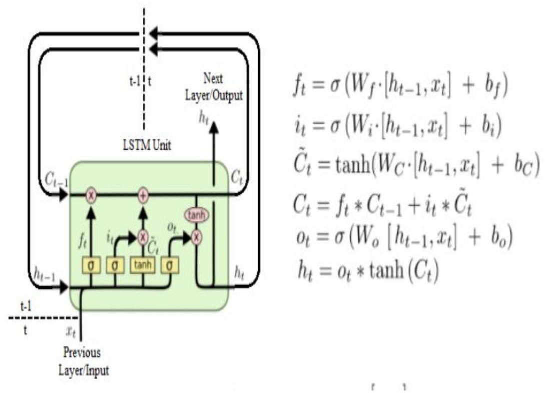

t-1 t
Next Layer/Output
h_t
LSTM Unit
C_{t-1}
C_t
f_t
i_t
$\tilde{C}_t$
o_t
tanh
σ
h_{t-1}
h_t
t-1
t
x_t
Previous Layer/Input
$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f)$
$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$
$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C)$
$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$
$o_t = \sigma(W_o [h_{t-1}, x_t] + b_o)$
$h_t = o_t * \tanh(C_t)$

Figure 3 LSTM Cell Architecture

The steps for this implementation is similar to the previous project. We create loaders for reading the train and test datasets. We also create a loader for reading the Word2Vec embeddings. The next step is to create an embedding dictionary(has features of size 300 for 1M words) for the words in the training dataset. If we the words are not present in the embedding dictionary we initialize it with zeros. The feature extraction step end with creating a tensor dataset and the dataloaders for the dataset.

The LSTM RNN network has three layers: the Embedding Layer , The bidirectional LSTM layer and a fully connected layer which is connected to Softmax function. We train the model with CrossEntropy as loss function and Stochastic Gradient Descent as optimizer. The dataset was trained in batches of size 32 and evaluated on the validation and test sets.

The detailed results and the observations from running the models is provided in the next section.

3. Experiment Result Analysis

As discussed in the previous section the model was run on the dataset containing 245821 tags in the dataset and for each tag is represented with approximately 78400 features. The model was run several times starting with small training sample size and eventually running it on the entire dataset. The model was run on Google Colab environment having 25 GB RAM. Despite of that the model could be trained, without batching for a maximum of 25000 samples in the training set. With batching the model could be trained for entire dataset. The metrics used for the classification report are accuracy, precision, recall and F1-score.Table 1 shows the accuracy report for the model without batching.

The test dataset contains around 40000 tags and when it was encoded with features of size 78400, the program ran out of memory. Hence the accuracy was calculated in batches of 10000 as well. Table 2 shows the accuracy report of the model when run on the entire dataset with batching.

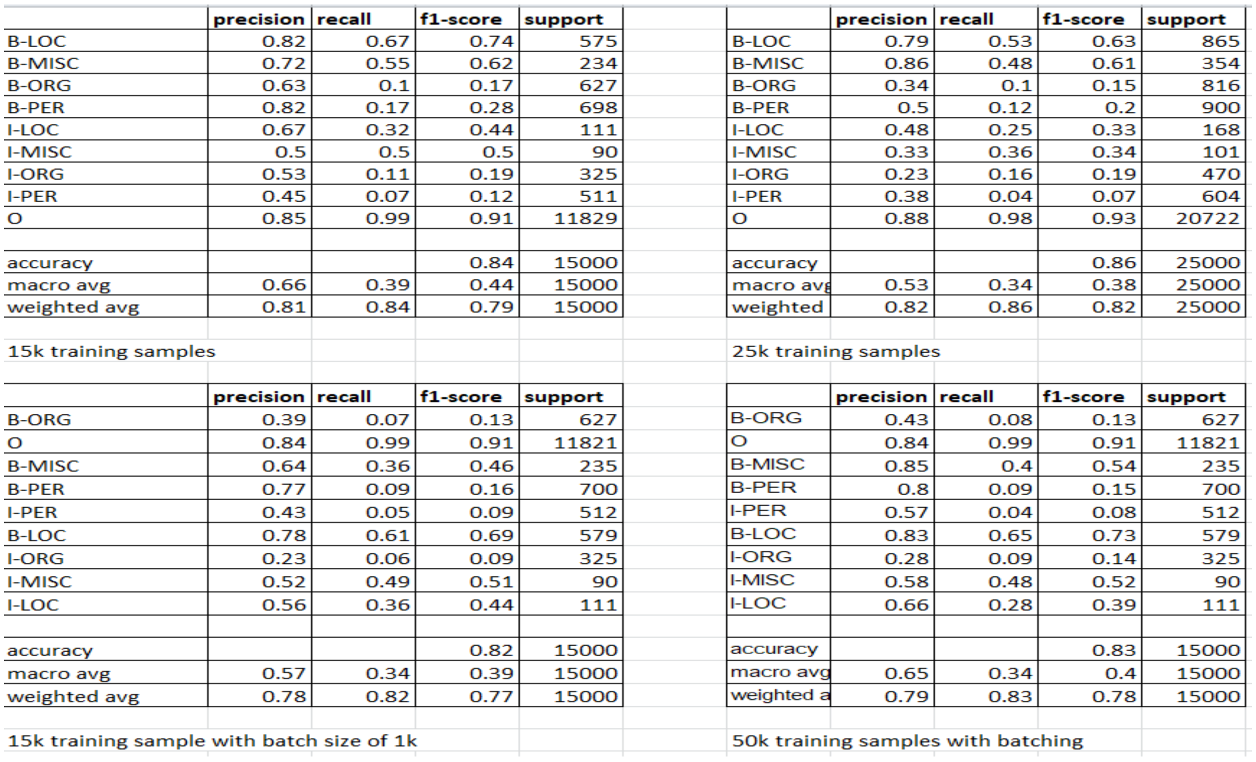
As evident from the table the general accuracy reported after running the model on large enough training sample is in range of 86-91%. On running the model with 10000 training samples with and without batching the accuracy dropped when batching was done. So maybe when batching is done increasing the number of iterations or randomizing the input sample can increase the accuracy of the model. Additionally from the table we can generally observe that the accuracy increased with increase in the number of training samples.

For the LSTM RNN deep learning model, deciding the loss function was a major challenge. With NLL loss the loss increased drastically after each epoch. Among NLL, CrossEntropy, MSE loss, CrossEntropy gave better result for consecutive epochs. While we got good results for Logistic Regression model, the accuracy was very low when we ran the model for limited dataset. (approx. 10000 training samples). The low accuracy may be attributed to the representation of multilabel output of the RNN which was used for calculating the classification report. The classification report when run for 9600 training samples is shown as below:

	prec.	recall	f1-score	support
1.0	0.00	0.00	0.00	395
2.0	0.00	0.00	0.00	3650
3.0	0.00	0.00	0.00	91
4.0	0.00	0.00	0.00	191
5.0	0.02	0.05	0.03	137
6.0	0.04	0.14	0.06	120
7.0	0.04	0.13	0.07	159
8.0	0.00	0.00	0.00	38
9.0	0.00	0.00	0.00	19
10.0	0.00	0.00	0.00	0
accuracy			0.01	4800
macro avg	0.01	0.03	0.01	4800
weighted avg	0.00	0.01	0.00	4800

Classification report for LSTM

Table 1 Logistic Regression Accuracy report without batching



	precision	recall	f1-score	support
B-LOC	0.82	0.67	0.74	575
B-MISC	0.72	0.55	0.62	234
B-ORG	0.63	0.1	0.17	627
B-PER	0.82	0.17	0.28	698
I-LOC	0.67	0.32	0.44	111
I-MISC	0.5	0.5	0.5	90
I-ORG	0.53	0.11	0.19	325
I-PER	0.45	0.07	0.12	511
O	0.85	0.99	0.91	11829
accuracy			0.84	15000
macro avg	0.66	0.39	0.44	15000
weighted avg	0.81	0.84	0.79	15000

15k training samples

	precision	recall	f1-score	support
B-LOC	0.79	0.53	0.63	865
B-MISC	0.86	0.48	0.61	354
B-ORG	0.34	0.1	0.15	816
B-PER	0.5	0.12	0.2	900
I-LOC	0.48	0.25	0.33	168
I-MISC	0.33	0.36	0.34	101
I-ORG	0.23	0.16	0.19	470
I-PER	0.38	0.04	0.07	604
O	0.88	0.98	0.93	20722
accuracy			0.86	25000
macro av	0.53	0.34	0.38	25000
weighted	0.82	0.86	0.82	25000

25k training samples

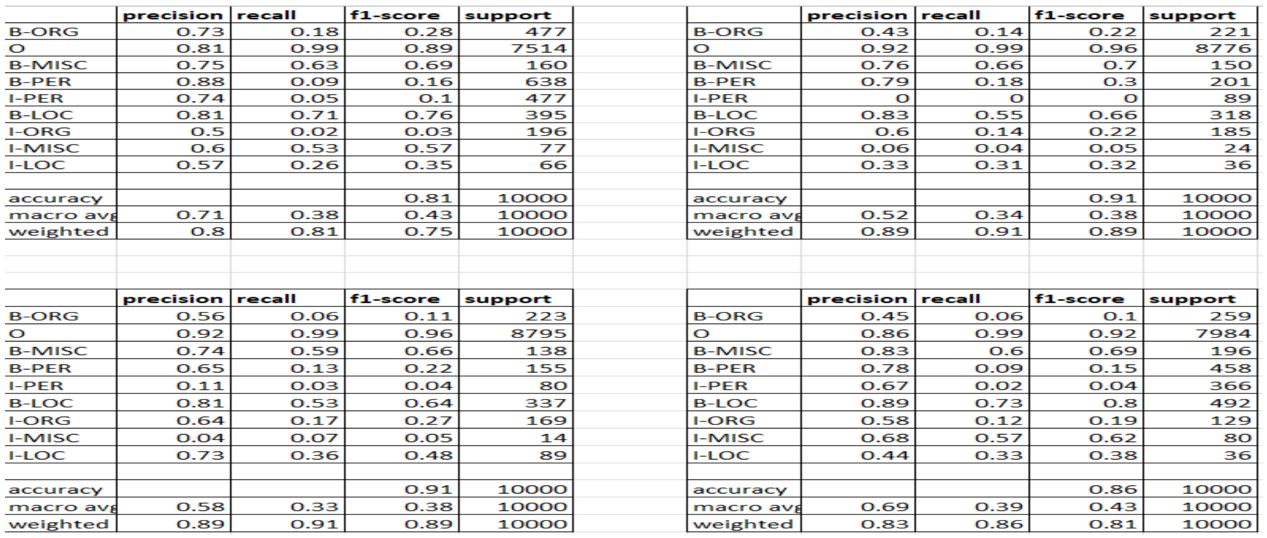
	precision	recall	f1-score	support
B-ORG	0.39	0.07	0.13	627
O	0.84	0.99	0.91	11821
B-MISC	0.64	0.36	0.46	235
B-PER	0.77	0.09	0.16	700
I-PER	0.43	0.05	0.09	512
B-LOC	0.78	0.61	0.69	579
I-ORG	0.23	0.06	0.09	325
I-MISC	0.52	0.49	0.51	90
I-LOC	0.56	0.36	0.44	111
accuracy			0.82	15000
macro avg	0.57	0.34	0.39	15000
weighted avg	0.78	0.82	0.77	15000

15k training sample with batch size of 1k

	precision	recall	f1-score	support
B-ORG	0.43	0.08	0.13	627
O	0.84	0.99	0.91	11821
B-MISC	0.85	0.4	0.54	235
B-PER	0.8	0.09	0.15	700
I-PER	0.57	0.04	0.08	512
B-LOC	0.83	0.65	0.73	579
I-ORG	0.28	0.09	0.14	325
I-MISC	0.58	0.48	0.52	90
I-LOC	0.66	0.28	0.39	111
accuracy			0.83	15000
macro avg	0.65	0.34	0.4	15000
weighted a	0.79	0.83	0.78	15000

50k training samples with batching

Table 2 Accuracy Report with Batching Entire Dataset



	precision	recall	f1-score	support
B-ORG	0.73	0.18	0.28	477
O	0.81	0.99	0.89	7514
B-MISC	0.75	0.63	0.69	160
B-PER	0.88	0.09	0.16	638
I-PER	0.74	0.05	0.1	477
B-LOC	0.81	0.71	0.76	395
I-ORG	0.5	0.02	0.03	196
I-MISC	0.6	0.53	0.57	77
I-LOC	0.57	0.26	0.35	66
accuracy			0.81	10000
macro avg	0.71	0.38	0.43	10000
weighted	0.8	0.81	0.75	10000

	precision	recall	f1-score	support
B-ORG	0.43	0.14	0.22	221
O	0.92	0.99	0.96	8776
B-MISC	0.76	0.66	0.7	150
B-PER	0.79	0.18	0.3	201
I-PER	0	0	0	89
B-LOC	0.83	0.55	0.66	318
I-ORG	0.6	0.14	0.22	185
I-MISC	0.06	0.04	0.05	24
I-LOC	0.33	0.31	0.32	36
accuracy			0.91	10000
macro avg	0.52	0.34	0.38	10000
weighted	0.89	0.91	0.89	10000

	precision	recall	f1-score	support
B-ORG	0.56	0.06	0.11	223
O	0.92	0.99	0.96	8795
B-MISC	0.74	0.59	0.66	138
B-PER	0.65	0.13	0.22	155
I-PER	0.11	0.03	0.04	80
B-LOC	0.81	0.53	0.64	337
I-ORG	0.64	0.17	0.27	169
I-MISC	0.04	0.07	0.05	14
I-LOC	0.73	0.36	0.48	89
accuracy			0.91	10000
macro avg	0.58	0.33	0.38	10000
weighted	0.89	0.91	0.89	10000

	precision	recall	f1-score	support
B-ORG	0.45	0.06	0.1	259
O	0.86	0.99	0.92	7984
B-MISC	0.83	0.6	0.69	196
B-PER	0.78	0.09	0.15	458
I-PER	0.67	0.02	0.04	366
B-LOC	0.89	0.73	0.8	492
I-ORG	0.58	0.12	0.19	129
I-MISC	0.68	0.57	0.62	80
I-LOC	0.44	0.33	0.38	36
accuracy			0.86	10000
macro avg	0.69	0.39	0.43	10000
weighted	0.83	0.86	0.81	10000

4. Summary and Conclusion

The goal of this project is to perform the task of NER tagging i.e. if we are given a sentence, the NER should extract all relevant named entities and assign each named entity with relevant tag. We started off with introducing the Named Entity Recognition problem in Natural Language Processing and briefly discussing the application areas of NER. It was followed by breaking down of the task to be involved in this project. We discussed each of the tasks in details. Finally we implemented different learning models and analyzed the results of training on these models.

As observed from the previous section, we got accuracy in range of 82-91% for the Logistic Regression model. One of the major roadblocks was running the model on entire dataset. The program crashed because of excessive memory usage. Using alternative approach for feature representation might help in mitigating this problem. This model was run on CoNLL based dataset containing sentences from news articles. As future work, these models can be trained on a variety of datasets having a mix of sentences from different contexts. With more architectural exploration and fine-tuning these models can achieve very good accuracies.

6. References

[1]https://medium.com/explore-artificial-intelligence/introduction-to-named-entity-recognition-eda8c97c2db1

[2] https://medium.com/@rohit.sharma 7010/a-complete-tutorial-for-namedentity-recognition-and-extraction-in-natural-language-processing-71322b6fb090

[3]https://towardsdatascience.com/named-entity-recognition-and-classi_cationwith-scikit-learn-f05372f07ba2

[4]https://towardsdatascience.com/named-entity-recognition-applications-and-use-cases-acdbf57d595e