Name Entity Recognition with Logistic Regression and Deep Learning

Ashutosh Agrawala
Net id – axa180037

Abstract

Named Entity Recognition, also known as entity extraction classifies named entities that are present in a text into pre-defined categories like individuals, companies, places, organization, cities", dates, product terminologies etc. It adds a wealth of semantic knowledge to your content and helps you to promptly understand the subject of any given text. In our project we try to train a model so that it can tag words in a sentence and categorize them into Person, Location, Organization and Miscellaneous. To train the model we use techniques like logistic regression and Bi-lstm model.

Introduction

Named entity recognition (NER)is probably the first step towards information extraction that seeks to locate and classify named entities in text into pre-defined categories such as the names of persons, organizations, locations, expressions of times, quantities, monetary values, percentages, etc. NER is used in many fields in Natural Language Processing(NLP), and it can help answering many real-world questions, such as:

- Which companies were mentioned in the news article?
- Were specified products mentioned in complaints or reviews?
- Does the tweet contain the name of a person? Does the tweet contain this person's location?

In this project we used a training dataset where each token in the sentence has been tagged with their corresponding tags. The tags that we are considering for this project are Person, Location, Organization and Miscellaneous. We build a bag of words model for our entire vocabulary in the given training dataset along with other features like pos-tags, lemma, hypernyms, hyponyms, holonyms and meronyms and build our feature space to train a logistic Regression Model which can be used to tag tokens in an unknown sentence. In the second part of the project I implemented a deep learning Model where I used a bi-lstm and assigned a class which is a ner-tag to each token in the input Sentence.

Steps:

In our pursuit to tag each of the sentences in the given input sentence I followed a certain series of steps involving feature extraction and model fitting.

Feature Extraction:

Step 1:

Found the length of the vocabulary by storing all unique words in a dictionary and then counted the length of the dictionary. The vocabulary size in the training set was close to 25000 words. This step is helpful in making a bag of words model for each of the given words in the training corpus.

Step 2:

The second step was to determine the pos-tag of each of the tokens in the sentence. This was accomplished using the nltk pos-tagger api which returns the most probable tag of each token in the given sentence. I also kept a track of all the **pos-tags** that were encountered as it would be used to concatenate to our bag of words model to build out feature space.

Step 3:

The third step in the process was to extract the **lemma** of each of the encountered words in the corpus so that the lemma if not present in the vocabulary could be added to the vocabulary list to expand the feature vector for each of the words.

Step 4:

The fourth step was to extract the **hypernym** of each of the encountered words if any take the 1st available hypernym of the word. If this word was not present in the vocabulary list it was added to the list to expand the feature vector of each of the words.

Step 5:

The fourth step was to extract the **hyponym** of each of the encountered words if any take the default hyponym of the word. If this word was not present in the vocabulary list it was added to the list to expand the feature vector of each of the words.

Step 6:

The fourth step was to extract the **meronym** of each of the encountered words if any take the default meronym of the word. If this word was not present in the vocabulary list it was added to the list to expand the feature vector of each of the words.

Step 7:
The fourth step was to extract the **holonym** of each of the encountered words if any take the default holonym of the word. If this word was not present in the vocabulary list it was added to the list to expand the feature vector of each of the words.

Logistic Regression Model Fitting:

Feature Vector Visualization:
Feature vector is vector of ones and zeroes. The way I made the feature vector for each of the word is as follows:

- Created a vector of zeroes of length equal to the vocabulary Length after adding all the extracted words in the feature extraction steps if the words were originally absent in the corpus
- For each of the word I assigned a unique integer using a simple counter
- Made the index corresponding to the integer associated with the word to be 1 and rest of the vector positions to be zero.
- For each of the extracted feature of the word that is the lemma, hypernym ,hyponym, meronym and holonym, if it existed and was a part of the vocab list we made the corresponding vector positions associated with the unique integer values of the model to be 1.
- This is what constituted our X_train for the Logostic Regression Model
- The Y_train for our Logistic Regression Model were the Ner-tags of each of the words in the form of an integer which represented a class for seen tags.

Fitting the Logistic Regression Model:

The Logistic Regression Model was trained using a newton-cg optimizer and our model was as it is one of the optimizers which supports multi-class classification. The default optimizer is lb-linear which only supports binary classification. The number of iterations for optimization was increased to 2000 iterations for our model to converge. The trained model was then used to predict the ner-tags of the Tokens in the sentence of the test set provided to us.

Results:

The analysis of our Model was done using the following metrics for each of our available tags/classes:

- Precision
- Recall
- F-score

Class	Precision	Recall	F-score
B-LOC	0.6060606	0.5976347	0.6018181
I-LOC	0.5609788	0.5743289	0.5675753
B-MISC	0.3307692	0.3678449	0.3483232
I-MISC	0.3476840	0.3847478	0.3652781
B-ORG	0.6947832	0.6693468	0.6818278
I-ORG	0.6621229	0.6784847	0.6702039
B-PER	0.7824994	0.7543276	0.7681552
I-PER	0.7263678	0.7845112	0.7543207
Overall	0.5837954	0.6014032	0.5946873

Conclusion

We can see that with proper feature selection and selecting defaults the logistic regression model is able to tag the tokens with an accuracy of approximately 65%. The model could perform better provided we use lisk algorithm for word sense disambiguation and more accurately extract the given features for the given set of tags in the corpus.

References

[1] https://towardsdatascience.com/named-entity-recognition-with-nltk-and-spacy-8c4a7d88e7da

[2] http://www.davidsbatista.net/blog/2018/05/09/Named_Entity_Evaluation/

[3] https://towardsdatascience.com/named-entity-recognition-and-classification-

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

[5] https://pythonprogramming.net/named-entity-recognition-stanford-ner-tagger/

[6] https://www.sicara.ai/blog/2018-04-25-python-train-model-NTLK-stanford-ner-tagger