Named Entity Recognition

Ansh Singhal
Union College
singhala@union.edu

Abstract

In this project we aim to create and evaluate two types of named entity recognizer for a Spanish Corpora; one that treats assigning tags to words as a classification task and the other involves a Maximum Entropy Markov Model in which we will also take into account the tag associated with the previous word while predicting the next tag in a sentence.

1 Introduction

Named Entity Recognition is a Natural Language Processing task of finding and classifying named entities in text. Unlike Part of Speech tagging, Named entity recognition utilises a smaller subset of tags to classify words. In this project, I will be creating two kinds of Named Entity Recognizers, one that simply treats assigning tags to words as a classification task, by extracting particular features from both the word as well as a window of words around it within the sentence to predict a label. For our second NER algorithm, we will be adding on to the algorithm that simply treats the task as classification, and utilising the previous tag in conjunction with the features to calculate a probability for each label given a word and the previous label. The labels given to each named entity is BIO encoded, which means that these labels are prefixed with a B or an I depending on whether the tag occurs in between of a larger entity or in the beginning of an entity (Daniel et al., 2007).

2 NER as a classification task

We made an NER by extracting features from each word in a sentence, then comparing those features to a golden label training set to fit a Logistic Regression classifier onto the trained feature and labels set. Deciding on a feature set that would maximize our F-score was the utmost priority at this stage, and so upon experimenting with various feature sets, here is the resultant f-score in each case:

Table 1: F-Score values for dev set for various feature combinations, for a window of one word around each word

	Features present	**F-Score**
$Test_1$	word	54.00
$Test_2$	word, capital	59.61
$Test_3$	word, capital, wordlen	57.05
$Test_4$	word, capital, punctuation	59.62
$Test_5$	word, capital, numeric	59.59
$Test_6$	word, capital, capitalization	60.20

In the end, I was thus able to select the features for each word as: for a 1 forward, 1 back window of three words for each word, 9 features are generated, that include the words, whether the first letter is capitalized and whether there are any numbers in the word. In this way, upon training and then classifying the words as named entities, I was able to get the following evaluation.

Table 2: Values of Precision, Recall, F-Score for the development file

	Precision (%)	**Recall (%)**	**F-Score**
Total	56.13	64.90	60.20
LOC	55.36	73.98	63.33
MISC	32.93	36.63	34.68
ORG	55.30	62.88	58.85
PER	67.08	70.70	68.84

Table 3: Values of Precision, Recall, F-Score for the test file

	Precision (%)	**Recall (%)**	**F-Score**
Total	60.66	69.51	64.78
LOC	67.86	71.68	69.72
MISC	29.69	31.27	30.46
ORG	57.54	69.79	63.07
PER	69.90	83.40	76.05

I simply excluded all the features which were detrimental to the f-score while keeping the features which boosted it.

3 NER as a sequence tagging task

Next we implemented our NER as a sequence tagging task, that is, we also take into account the previous tag while deciding upon the probability of the tag for the current word. We did so using the Maximum Entropy Markov Model, thus the new feature set involves a framed slice of the words, capitalization of said words, and whether or not said words contain a number respectively as well as the tag of the previous word (Defaulted to '<S>' for the previous tag for the first word in a sentence). We trained a Logistical Regression Classifier on these features and then used a Viterbi function to predict the most probable sequence of tags.

The Viterbi decoding algorithm keeps track of each possible state (tag) and the most probable path (*sequence of tags*) to get there. For each word, we look at the probability of each label given the features of the word as well as the previous tag, and in this way calculate a 'Viterbi' value for each possible label, and carry forward this process for every word. In the end, we select the final state that has the highest probability, and simply return the $path$ of tags that got us there. In doing so, we get a predicted list of tags which are treated as a sequence. These were the results from the viterbi Named Entity Recognizer when ran on the Spanish Corpora development and test sets:

Table 4: Values of Precision, Recall, F-Score for the development file

	Precision (%)	**Recall (%)**	**F-Score**
Total	73.62	69.64	71.57
LOC	64.41	72.46	68.20
MISC	59.72	48.31	53.42
ORG	78.82	71.35	74.90
PER	80.09	72.75	76.24

Table 5: Values of Precision, Recall, F-Score for the test file

	Precision (%)	**Recall (%)**	**F-Score**
Total	77.03	74.85	75.92
LOC	80.55	72.97	76.57
MISC	65.07	43.95	52.46
ORG	75.56	79.07	77.28
PER	78.87	83.81	81.27

Conclusion

Treating the task of named entity recognition clearly boosted the performance of the model. It is obvious, that in the task of named entity recognition, and part of speech tagging in extension, there are certain patterns when it comes to the sequence of tags, and the viterbi decoder picks up on these signals, leading to a higher overall F-score. One area of major improvement was the 'Miscellaneous' tag, which although has fewer occurrences in the whole corpora, performed really poorly, since the features were not a good enough indicator for 'Miscellaneous' words. However, they do occur in specific patterns within a sentence, and the model that treated the task sequentially was able to drastically improve the performance with regards to this label in particular.

References

Jurafsky Daniel, Martin James H, et al. 2007. *Speech and language processing: An introduction to natural language processing, computational linguistics, and speech recognition.* prentice hall.

I affirm that I have carried out my academic endeavours with full academic honesty