Named Entity Recognition

Claudia Porto

Abstract

In this project we are implementing two Named Entity Recognizers (NER). The first one treats NER as a classification task, while the second one uses a classifier that takes into account the tag predicted for the previous word. The second NER model uses the Viterbi algorithm to determine the most likely sequence of tags for a given input sentence. We are looking at the performance of each NER and how the performance metrics of the second NER are affected when including the last word's predicted tag.

1 Introduction

Named Entity Recognition is the task of finding and classifying named entities in text. NER usually uses a relatively small number of tags, where the vast majority of words are tagged with the "non-entity" tag, or O tag. In this project, we implented two versions of a named entity recognizer. The first treats NER as a classification task, and the other will also use a classifier, but also takes into account the tag predicted for the not only the current word, but the previous word as well. The second NER utilizes the Viterbi algorithm, and is known as a Maximum Entropy Markov Model (MEMM).

2 Data

The data used in this project comes from the Conference on Natural Language Learning(CoNLL) 2002 for Spanish and Dutch. We will be using the Spanish corpus in this project. The data uses BIO encoding, meaning that each named entity tag is prefixed with either B- (beginning), or I- (inside). The other tags include PER (person), LOC (location), ORG (organization), MISC (miscellaneous), and O (non-entity).

3 Evaluation

There are two common ways to evaluate NER systems. Those of which include both phrase-based and token-based. In our project, we use phrase-based to report scores. This means that a system must predict the entirety of the phrase span correctly for each name.

4 NER as a Classification Task

In our first NER used as a classification task, we are using a logistic regression classifier from sklearn. We then specify our features as name-value pairs and use sklearn's DictVectorizer to turn the feature name-value pairs into a vector of number that the classifier can utilize. We began with an implementation with a classifier that gets an F1 measure of over 50 percent right from the start. It was our task to evaluate and add features to raise that F1 evaluation measure.

4.1 Features

The features used in my final implementation of the first NER are below:

'word': returns the word itself

'isCapitalized': returns 'True' if the first character of the word is uppercase and 'False' otherwise

'isDigit': returns 'True' if the word is composed entirely of digits and 'False' otherwise

'isAllCaps': returns 'True' if all characters in the word are uppercase and 'False' otherwise

'isAllLower': returns 'True' if all characters in the word are lowercase and 'False' otherwise

'isPunct': returns 'True' if the word consists of a single non-alphanumeric character (e.g., !) and 'False' otherwise

Evaluation Metric	Value(%)
Accuracy	97.01%
Precision	70.42%
Recall	74.91%
F1	72.60%

Table 1: Overall Evaluation Metrics for NER as a classification task

LOC	Value(%)
Precision	70.24%
Recall	80.82%
F1	75.16%
MISC	**Value(%)**
Precision	61.11%
Recall	47.83%
F1	53.66%
ORG	**Value(%)**
Precision	71.07%
Recall	74.14%
F1	72.57%
PER	**Value(%)**
Precision	72.13%
Recall	80.00%
F1	75.86%

Table 2: Evaluation Metrics for Each Tag

'hasHyphen': returns 'True' if the word contains a hyphen character ('-') and 'False' otherwise

'isIdentifier': returns 'True' if the word is a valid Python identifier (i.e., it consists only of letters, digits, and underscores, and does not begin with a digit) and 'False' otherwise

'hasApostrophe': returns 'True' if the word contains a apostrophe character (''') and 'False' otherwise.

Other features were added and then removed such as 'wordLength', 'prefix', 'suffix', and also specific prefixes and suffixes such as -ing, -s, re-. These features were removed due to the nature of the data. These features were primarily specific to the English language, and would be of no help to our classifier, because we are using a Spanish corpus.

Table 1 shows the overall evaluation metrics for the NER classifier. We can see that by adding these additional features to the implementation, the F1 score measure increased by about 20% from its original 50% F1 score to begin with. Table 2 shows the evaluation metrics for each specific tag. We can see that PER had the best performance based on the additional features and MISC had the lowest performance. Overall, the inclusion of these features, although very generalized, made a big impact on the performance of the classifier. If we were to add more features specific to Spanish, I would expect for the performance to be even greater.

5 NER as a Sequence Tagging Task

In our second NER used as a sequence tagging task, we are using a logistic regression classifier from sklearn. In the second part of our project, we implemented a Maximum Entropy Markov Model (MEMM) with the Viterbi algorithm to predict named entities in text. In contrast to the first part, where we treated NER as a classification task, the second part involves utilizing a classifier that takes into account the tag predicted for the previous word. By incorporating this contextual information, we can improve the overall accuracy of our NER system.

5.1 Features

The feature set used in the second part of our project is similar to the first feature-wise, with the same reasoning as to why features were included or discarded. However, we also include information about the previous tag in our feature set to capture contextual information. Specifically, we include the following features:

'word': returns the word itself

'isCapitalized': returns 'True' if the first character of the word is uppercase and 'False' otherwise

'isDigit': returns 'True' if the word is composed entirely of digits and 'False' otherwise

'isAllCaps': returns 'True' if all characters in the word are uppercase and 'False' otherwise

'isAllLower': returns 'True' if all characters in the word are lowercase and 'False' otherwise

'isPunct': returns 'True' if the word consists of a single non-alphanumeric character (e.g., !) and 'False' otherwise

Evaluation Metric	Value(%)
Accuracy	97.49%
Precision	82.63%
Recall	80.15%
F1	81.37%

Table 3: Overall Evaluation Metrics for NER as a sequence tagging task

LOC	Value(%)
Precision	80.26%
Recall	83.56%
F1	81.88%
MISC	**Value(%)**
Precision	81.25%
Recall	56.52%
F1	66.67%
ORG	**Value(%)**
Precision	85.59%
Recall	81.90%
F1	83.70%
PER	**Value(%)**
Precision	80.36%
Recall	81.82%
F1	81.08%

Table 4: Evaluation Metrics for Each Tag

'hasHyphen': returns 'True' if the word contains a hyphen character ('-') and 'False' otherwise

'isIdentifier': returns 'True' if the word is a valid Python identifier (i.e., it consists only of letters, digits, and underscores, and does not begin with a digit) and 'False' otherwise

'hasApostrophe': returns 'True' if the word contains a apostrophe character (''') and 'False' otherwise.

Table 3 shows the overall evaluation metrics for the NER classifier as a sequence tagging task. We can see that including the last word's predicted tag increased the F1 performance of about 70% to about 80%. By including this, it has provided the classifier more necessary information when tagging each word in the sentences, and a big leap in performance is seen. Table 4 shows the evaluation metrics for each specific tag. We can see here that ORG had the best performance based on the additional features, compared to PER in the first NER classifier, and MISC had the lowest performance once again. Overall, the inclusion of these features and including the last word's predicted tag, made an even bigger impact on the performance of the classifier. Again, if we were to add more features specific to the Spanish language, I would expect for the performance to be even greater.

6 Conclusion

In this project, we implemented two named entity recognizers (NER) using the CoNLL 2002 Spanish corpus. The first NER treats NER as a classification task, utilizing a logistic regression classifier from sklearn. The second NER is a Maximum Entropy Markov Model (MEMM) that uses a classifier taking into account the predicted tag for both the current and previous words. The Viterbi algorithm is used to determine the most likely sequence of tags for an input sentence.

We evaluated our models using phrase-based scoring, which requires the system to predict the entire phrase span correctly for each name. We utilized various features to improve the performance of our models.

Overall, our second model, the MEMM, outperformed the first model, achieving an F1 score of over 80%. The inclusion of the previous word's predicted tag was found to have a positive impact on the performance of the MEMM. Our project demonstrates the effectiveness of utilizing MEMMs for NER tasks and highlights the importance of carefully selecting and engineering features to improve performance.

7 Honor Code

I affirm that I have carried out my academic endeavors with full academic honesty. Claudia Porto