Mini 1 Named Entity Recognition Report

Chenkuan Liu
Natural Language Processing
UT Austin
Jan 2021

1 Introduction

This mini project implements a simple classifier for named entity recognition. Namely, given a list of tokens, it can determine whether each token inside is part of a person's name. In general, the program has the following procedure: dictionary generation, feature extraction and vectorization, gradient descent training, and final prediction.

1.1 Method

The first step is to generate an integer-to-string dictionary from the training set such that each token in the training data is associated with a unique number. And then, I set up 12 features such that each token is extracted and expanded into a sparse feature vector based on them. After that, I initialize the weights as a zero vector and use logistic regression with adaptive gradient descent to train on the data to update the weights. After sufficient epochs, the weights become stabilized. Then, during testing, each token in development set is extracted into a feature vector based on the same standard as the training tokens. These feature vectors will be multiplied with the trained weights. After that, I add an optional step that newly encountered tokens with specific properties will be given additional weights to increase their chance of being part of a person's name. This step will be discussed in further detail. After the value of $w^t x$ is computed, I will calculate its logistic. If the resulting value is greater than or equal to 0.5, it will be predicted as part of a person's name, otherwise it will not.

1.2 Features

The 12 features can exchange their orders as long as they remain consistent during training and testing. These features are:

Proper Capitalization This feature returns 1 if the first character of the token is capitalized but not all characters in the token are capitalized. The logic behind this feature is that the first letter of a person's name is typically capitalized, but there are situations such as news headlines where all letters are capitalized and so we need to exclude those tokens.

Beginning Position This feature returns 1 if the token is at the beginning of the list.

"Upper Dot Format" This feature returns 1 if the token has the form equivalent to "A.", namely a capitalized letter followed by a dot. The reason is that such format typically represents the middle name of a person.

Previous "Upper Dot Format" This feature returns 1 if the previous token is of the "Upper Dot Format" as defined above.

Token Length This feature returns 1 if the token has length less than three and is not of the "Upper Dot Format" defined above.

Digits This feature returns 1 if the token contains digits.

Capitalization in previous and next tokens This feature returns 1 if both the previous and next tokens are Capitalized. For token at the beginning, it will only check the next token; for token at the end, it will only check the previous token.

Previous, Current, and Next Tokens This collection of features returns the indices of the token before the previous token, previous token, current token, next token, and the token after the next token. These are five features in total.

1.3 Additional Step during Prediction

For comprehensiveness, I used all tokens in training set to generate the dictionary. Therefore, during training stage, the situation where a token is not inside the dictionary will never appear. However, when testing on development set, there will be new tokens outside of the dictionary, and a certain amount of them will be persons' names that never encountered in training set. To take account of this, after the weighted score $w^t x$ is computed, I added an additional step: if the token is not in the training dictionary, has length more than 3, and is properly capitalized, then its score will be added by an additional 1.25. This is an optional step and the value 1.25 is set empirically. Next section will examine the performance of the method with different epoch numbers and whether or not this additional step is included.

2 Performance

The optimizer used is unregularized adaptive gradient descent with zero weight initialization and 1.0 learning rate. The total epoch numbers experimented are 30 and 50. Their results are:

Table 1: Plain developing set results

Epochs	Accuracy	Precision	Recall	F1-score
30	0.9893	0.9492	0.8717	0.9088
50	0.9896	0.9504	0.8762	0.9118

Table 2: Developing set results with additional step in 1.3

Epochs	Accuracy	Precision	Recall	F1-score
30	0.9900	0.9259	0.9082	0.9170
50	0.9902	0.9284	0.9101	**0.9192**

We can observe that in both cases F1-score increases as we train for more epochs. Using plain prediction, we reach 91.2% F1-score with 50 epochs. However, if we include the additional step described in section 1.3, the F1-score will become 91.7% with 30 epochs and 91.9% with 50 epochs. Furthermore, observing the predictions and recalls, we can see that, since we added 1.25 to all tokens satisfying relevant conditions during prediction, there will be some tokens that are not persons' names but satisfied such conditions and get assigned to 1. As a result, it will lead to increases in recall but slight decreases in precision, as suggested in table 2. The configuration with the highest F1-score (0.9192) is used to predict and write the labels into the blind test set `eng.testb.out`.