CS388 Mini 1: Classification for Person Name Detection

Yixuan Ni, yn2782

Abstract

In this project, I implement a logistic regression classifier for labeling person names in a sentence. The best model achieves an F_1 score of 91.53. I also evaluate and rationalize the effects of various features on the results.

1 Introduction

In this project, we are required to implement a classifier that identifies whether a token in a given sentence is a person's name. The data for this project is part of the CoNLL 2003 Shared Task on Named Entity Recognition, labelled with named entities types: person, organization, location, and miscellaneous. In this project, we keep the person label and formulate the training data so that if a token is part of a person's name, it is labeled 1; otherwise, it is labeled 0. The goal of the classifier is, when given an unlabeled sentence, predicting the label for each token in the sentence.

2 Method

For this project, I built a logistic regression classifier. I extracted and experimented with the features listed in Table 1. The features are extracted from the current token, as well as the context of the current token (previous and next). I used a sparse representation of the features: the features all take binary values and are indexed. A list of indices of active features are kept for each token.

For training, I used Unregularized Adagrad Trainer with η of 1.0. The weights were initialized to be 0. I trained each model for 10 epochs and kept the model with the highest F_1 score on the development set.

To choose the best feature combination, I first built a benchmark with all of the features. Then I removed each feature one-by-one until the highest F_1 score was achieved.

3 Results

Table 2 includes some of the experimental results on different features. These results provide insights on how each feature affects the model predictions. Only using the current token as feature gives a pretty good baseline of $F_1 = 77.02$. The word itself seems to provide most of the information for fitting the model.

[1]Shape is a result of mappings like Windows7 to Xxxxxxd (upper letter to X, lower letter to x, digit to d).

Feature Name	Details
token	index of the token in the vocabulary
isPerson	whether it was labeled PER in train set
pos	pos tag of the token
posPrefix	prefix of the pos tag of the token
prefix	prefix of the token
suffix	suffix of the token
isUpper	whether it begins with upper letter
isDigit	whether it consists of all digits
index	index of the token in the sentence
len	length of the token
shape[1]	shape of the token
BOS	whether it is the begin of the sentence
EOS	whether it is the end of the sentence

Table 1: Features extracted for each token. These features are extracted from the current token, as well as the context of the current token (previous and next). The combination that gives the best result is: **token, token (lower), pos, prefix, suffix, isUpper, shape, previous token (lower), next token (lower), and next pos**.

Features	F_1
current all + context all	0.8161
current token	0.7702
current token + context tokens	0.7854
current all + previous all	0.8098
current all + next all	0.7715
best	**0.9153**
best w/t pos	0.9070
best w/t token	0.9085

Table 2: Experiment results. Not all results are included due to page limit.

Features extracted from the previous token has larger effect than those from the next token.

Having more features does not produce a better model. Too many features makes the model overfit the training set. The model with all features has $F1 = 0.9922$ on the training set, but only 0.8161 on the development set.

The best feature combination is: token, token (lower), pos, prefix, suffix, isUpper, shape, previous token (lower), next token (lower), and next pos. The best F_1 score achieved is 0.9153. Adding each feature does not increase the score much. It is the combination that matters the most comparing to individual features.