Classification for Person Name Detection

Rose Zhang

Abstract

This project involved experimenting with how different features affect the detection of names of people in logistic regression.

1 Introduction

Using a dataset of sentences that are labeled as either a Person's name or not a name, I used logistic regression to train a model to recognize which words in a sentence are names. I came up with a set of features to train the model. Sparse features with yes/no answers like whether a word is capitalized or not are used in feature extraction, so sparse features can take in 1 or 0 depending on if the feature is present. In this project, I used capitalization, length of word, number of appearance in a sentence, if it was the first or last word, the word before or after the current word, and multiple representations of the word: the word itself, the word in lower case, the first and last three characters of the word, and the shape of the word, where shape is whether each character in the word is capitalized, lower case, a number, or special character. The SGD optimizer was used. The model is then trained on a training set and evaluated on a dev set.

2 Results

FEATURE	F1 WITHOUT
CAPITALIZED	88%
PREV/NEXT WORD	87%
FIRST/LAST 3 CHAR	89%

Table 1

In Table 1 are some notable features in the model and how they affect the F1 score of the dev set when they are left out. The F1 score with all features is 91%. It makes sense that capitalization and the previous and next word features have the strongest influence. Names are always capitalized and are often denoted by a prefix like Mr. or Dr. before or a Jr. after. One name token can follow another name token in full names. And names with common roots can share common beginnings and endings which made using the beginning and end of the name useful. Additionally, reducing to only one epoch from 4 reduces F1 to 86%.

I also tried several features that had no effect. Checking if the token was a number or contained numbers didn't help, presumably because those tokens were filtered out already by other features that were used. I also tried compressing the shape by combining characters in the shape that were repeating, for instance, the shape of "Adam" would be "Cccc" which would be compressed into "Cc". However, this actually worsened the F1 score slightly. Perhaps because the compressed shape became too generic and there was no distinguishing compressed shape for a name. Dividing the length features into smaller groups (e.g. splitting the <=5 category into <=3 and 4-5 categories) also didn't help. Splitting further didn't help differentiate whether a longer or shorter word in the subgroup was a name or not and was too specific to be useful. Finally, increase the epochs too much decreased the F1 score and worsened performance on the dev set while it helped immensely with the training set. This shows that using a smaller number of epochs on this small dataset is better to avoid overfitting.