COMP9318 Data Warehousing and Data Mining

Project Report

Nijie Sun
z3447959

Table of Contents

Introduction 3
1. Task 3
2. Implement: trainer and tester 3
Choose features 3
1. Method and values to evaluate features 3
2. Features 4
Experiment and improve classifier 7
1. Regularization 7
2. Improve classifier by cross-validation and regularization 7
3. Final parameters decision 11

Introduction

1. Task

The task of is project is to perform NER(Named Entity Recognition) for a single category of TITLE by using LogisticRegression classifier in scikit-learn. Logistic regression use formula w^T *X = y, X is the input data, and y is the category result. By using known category data, try to learn the parameter w and get more accurate predict y.

2. Implement: trainer and tester

a) Initialize data

Step1: add features

The training data format is: [word, tag, category]. Since word and speech tag are chosen as features in my trainer and tester (will explain later), new features are created as columns insert between tag and category. Add all the columns except category to X, category to y.

Step2: prepare X and y for logistic regression

Use pandas to_dict() function and DictVectorizer fix_transform() function to victories X, so that scikit-learn LogisticRegression can use. This is because elements in X need to be numbers, so that we can calculate w^T *X to find the best weight vector.

b) Logistic Regression

In trainer, use logistic regression to training classifier, input the training data X, and category (‘TITLE’ or ‘O’) vector y, and try to find the proper weight w and save into classifier file.

In tester, use the classifier trained to predict y.

The goal of training is to train a classifier which can predict more accurate y

Choose features

1. Method and values to evaluate features

Cross-validation is normally used to judge a model. We use K-fold cross validation to get a reliable estimate of how well a model performs on unseen data (i.e., the generalization error). This helps to (1) determine which model works well, or (2) how to set the values for the hyper parameters for a model. This part it is used for (1), that is whether a feature is good for a model. Later it will be used to (2), set hyper parameter.

When we only have training data set, we need to know the test accuracy, we can separate training data set into k parts. Each time use one part as test data, others as training data Train the classifier by training data and score both training data and test data. If the accuracy of predicting training data is very high but the accuracy of predicting test data is very low, the model is overfitting. If the accuracy of predicting training data and test data are both very low, the model is underfitting.

Since f1 value is used to evaluate the classifier in this project, average f1 value is calculated in the trainer as well.

I use two f1 values to evaluate a feature is good or not. The first one is the average f1 value of cross-validation. In this stage, I set all the parameters of LogisticRegression() as default. The second one is the f1 value of testing the whole training data set by using trained classifier.

Since the f1 values of cross-validation only has slightly difference (normally no more than 0.01), the f1 value of testing the whole training set is used more often. So the following f1 value to describe features in next part means the f1 value of the whole training set.

2. *Features*

a) Word and tag

Feature describe:

The word itself is used as the only feature first. It is because it has more information than only use tag. I get f1=0.878333333333 by using word itself as a feature.

When I also add tag as the second feature, the f1 is 0.890354492993(increase about 0.012).

Why choose:

These two column are from original data, they contain the original information. They both should be put into features except they will have negative affect to the result. The tag can also increase the f1 value is reasonable. For example, a word with speech tag 'CONJ' or 'ADJ' is impossible a title. Of course the tag has more information than this.

b) Word and tag of previous and next word

Feature describe:

The previous two and next two words and their tags are used as feature columns. If the word itself is the first two or last two words in a sentence, the corresponding column will be set to special character.
the f1 value is 0.9238005559528906 when I add these previous and next words and tags. These features contribute about 0.033 f1 value. This is a good improvement.

Why choose:

The previous two and next two words are used as features because there are some words that are frequently used together as title and all of them are set 'TITLE', or some words are used together and the first or second one will be title. I hope classifier could learn some information about it.

There previous two and next two tags are used as features because the title will have some combination of tag types with previous and next words. For example, a title is often has word with tag 'NNP' after it as name or VB after it to describe this person do something. If the classifier could learn these kind of information, the predict accuracy will be improved.

c) Gazatteer

Feature describe:

Gazatteer is a method to improve accuracy by give classifier the information of a list of known title. My trainer and tester simply add a feature column indicate weather this word is in known title list or not. Set '1' to this column if this word is in the known title list, '0' otherwise.

This feature change f1 from 0.91015625 to 0.941224938204 (increase about 0.03)

Why choose:

This feature is expected to be useful for the test set in other domain. It can also get 0.03 increasing of f1 when testing training data. It is also good improvement.

d) Previous and next words in known title list or not

Feature describe:

Similar as Gazatteer, just use two feature columns in X to indicate the previous word is in known title list or not and the next word is in known title list or not.

It is amazing that this feature change f1 to 0.987852089174 (increase about 0.0466)

Why choose:

This feature is set to capture the information that some words that are frequently used together as title and all of them are set 'TITLE'. 'Chief Executive Officer' is an example of this type. If the next word 'Officer' is in the known title list , then the probability of 'Executive' be predict as a 'TITLE' will increases.

e) Transform plural word to singular and

Feature describe:

This feature converts a word to singular mode if it is plural as a new column in X. It only increase f1 about 0.003.

Why choose:

Plural words are seldom used as a title. This is the information I expect classifier to learn about. It is not a strong feature, so the effect is slight.

f) First character is upper case

Feature describe:

Set the column to '1' if the first character of the word is upper case, '0' otherwise.

Why choose:

The first character of some titles is upper case. 'Prime' 'Minister' and 'Mr' are examples. I hope this feature will be useful with the combination of other features. However, the effect is smaller than converts a word to singular. This feature still be used is because it can increase f1 of one other domain test data set.

Experiment and improve classifier

1. Regularization

Regularization is a method to deal with overfitting. It is designed to penalize large values of the model parameters. Hence it encourages simpler models, which are less likely to overfit. We can choose L1 regularization (Lasso LR) or L2 regularization (Ridge LR). L1 regularization is more likely to result in sparse models. We can also choose proper hyper-parameter to control the strength of regularization.

2. Improve classifier by cross-validation and regularization

After features have been decided, the next task is to improve classifier by finding the proper parameters of sklearn LogisticRegression(). That is find proper regularization parameters. Parameter 'Penalty' can be set as 'l1' or 'l2' and 'C' is a positive float. 'C' is the Inverse of regularization strength, smaller values specify stronger regularization. It has been mentioned in features part that cross-validation can also be used to find proper values for the hyper parameters for a model. Cross-validation are used in all the following experiments.

Since it is not clear whether a sparse model is better in this project, I tried both 'l1' or 'l2'.

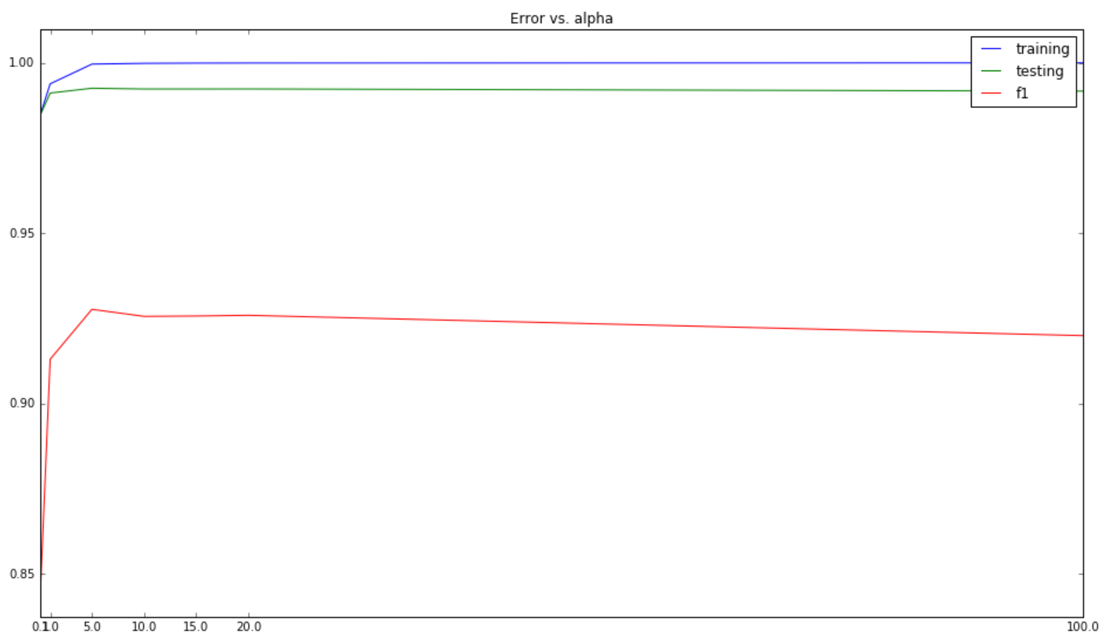
'l1' is set as 'Penalty' first. In fact I tried C in [0.001, 0.01, 0.1, 1, 10, 100] first, It is better to plot and find the best value range of C so I put the result of C in [0.1, 1, 5, 10, 15, 20, 100] here.

a) Penalty = L1, C in [0.1, 1, 5, 10, 15, 20, 100]

C	avg. training accuracy	avg. testing accuracy	avg. f1
0.1	0.9851780825835481	0.9849399431752699	0.8473874314881936
1	0.9937923425730982	0.9910880613104494	0.9129440464967795
5	0.9995785311290343	0.9924981564693229	0.927603665485301
10	0.9998088504894532	0.9922725501523463	0.9255347858446173
15	0.9998762227736145	0.9922725421983793	0.9256434711050721
20	0.9998981579484594	0.9922866505471573	0.9258445376096148
100	0.9999482956557875	0.9916520969878089	0.9198813868238777

Table 1 Penalty = L1,

Penalty = L1, C in [0.1, 1, 5, 10, 15, 20, 100]

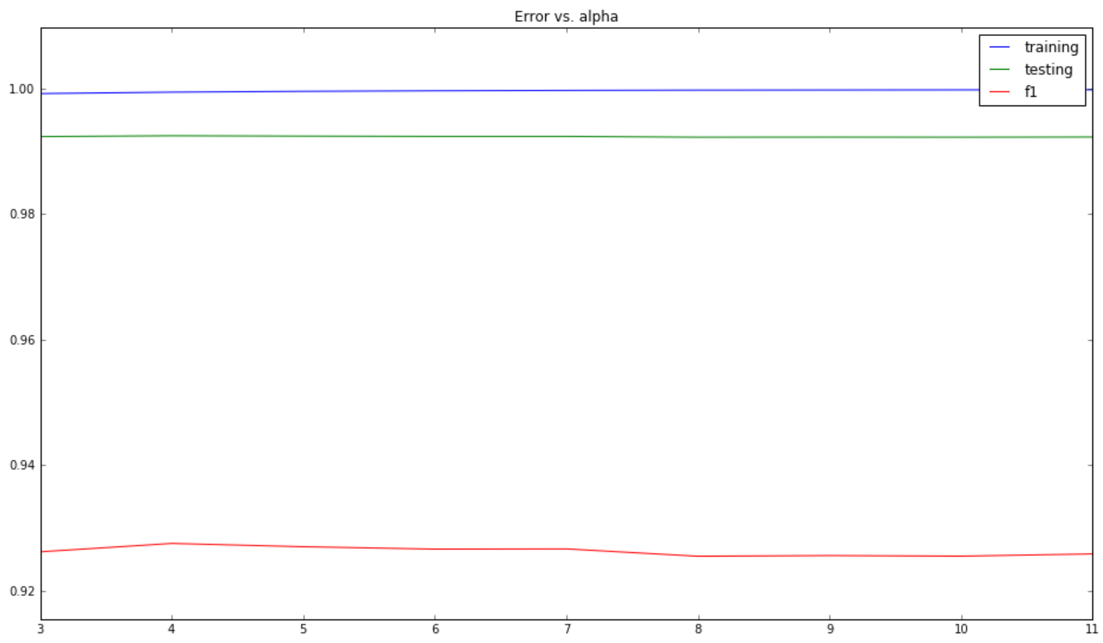


Graph 1 Penalty = L1, C in [0.1, 1, 5, 10, 15, 20, 100]

According to Graph1, It is clear that the regularization strength C which get the highest f1 value is the same as the C in which the avg. testing accuracy is the highest and closest to the avg. training accuracy. That means the best C value to avoid overfitting also gets the highest f1 value. We can find same result in later experiments. So we only need to care about f1 now. Here 5 is the best C value.

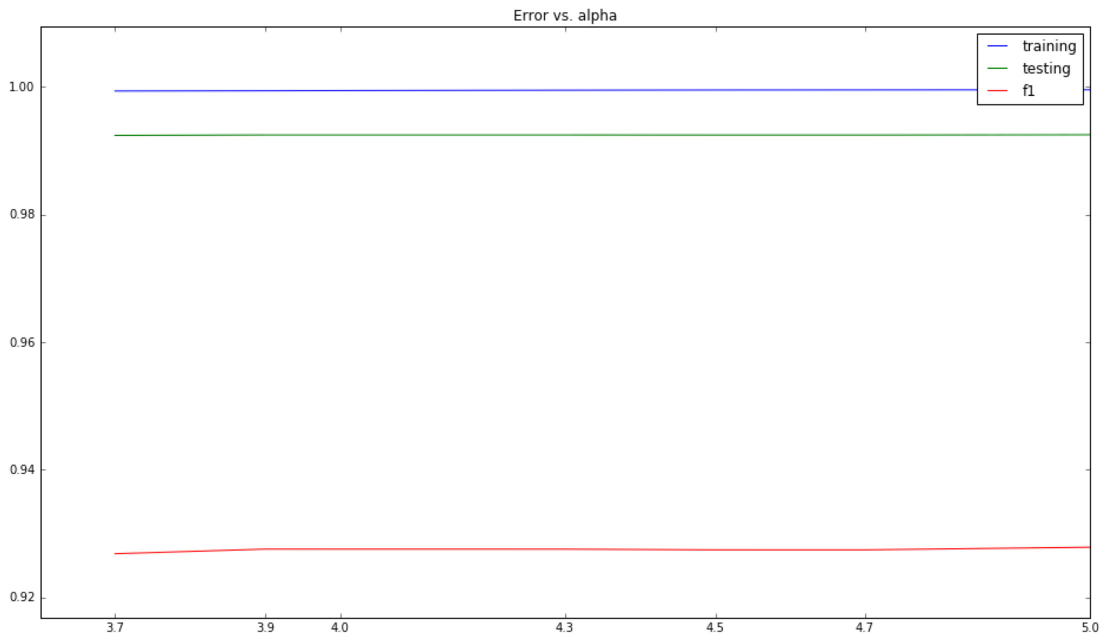
b) Penalty = L1, C in [3, 4, 5, 6, 7, 8, 9, 10, 11]

We change C to a small range to get more accurate C value that get the highest f1. According to Graph 2, it is clear that C=4 get the highest f1.



Graph 2 Penalty = L1, C in [3, 4, 5, 6, 7, 8, 9, 10, 11]

c) Penalty = L1, C in [3.7, 3.9, 4.0, 4.3, 4.5, 4.7, 5]

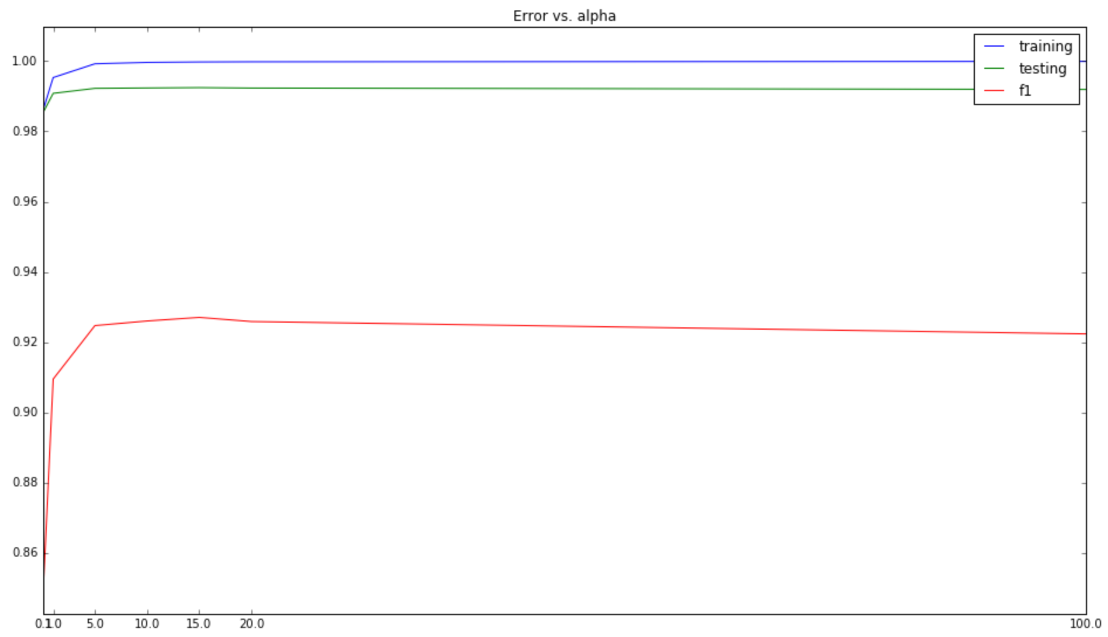


Graph 3 Penalty = L1, C in [3.7, 3.9, 4.0, 4.3, 4.5, 4.7, 5]

We change C to a much smaller range to get more accurate C value that get the highest f1. According to Graph 3, it is clear that C=3.9 get the highest f1 = 0.9275814971701747. So C=3.9 will be chosen if use penalty=l1.

d) Penalty = L2, C in [0.1, 1, 5, 10, 15, 20, 100]

Now l2 is used.

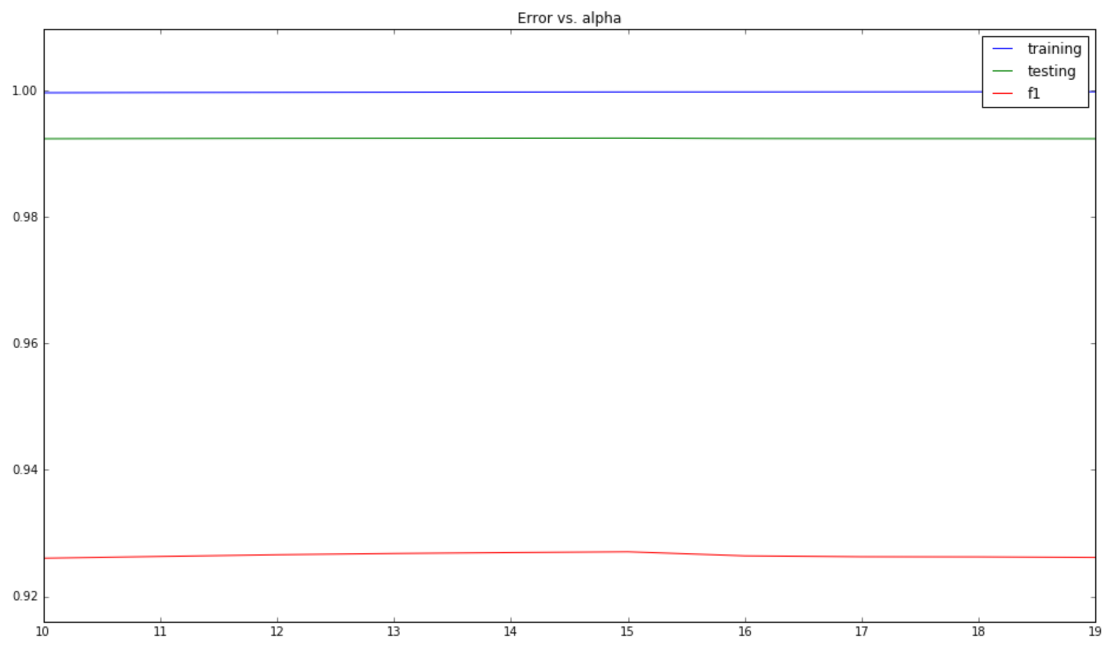


Graph 4 Penalty = L2, C in [0.1, 1, 5, 10, 15, 20, 100]

According to Graph4, C=5 get the highest f1 value.

e) Penalty = L2, C in [10, 11, 12, 13, 14, 15, 16, 17, 18, 19]

We change C to a small range to get more accurate C value that get the highest f1. According to Graph 5, C=15 still get the highest f1 = 0.9270537320776848.



Graph 5 Penalty = L2, C in [10, 11, 12, 13, 14, 15, 16, 17, 18, 19]

We find (penalty=l1, C=3.9) get f1=0.9275814971701747;

(penalty=l2, C=15) get f1=0.9270537320776848

There is only a small difference between these two group of parameters.

3. *Final parameters decision*

I tried several test data sets in other domains and find (penalty=l2, C=15) get better result of f1 values although it gets slightly lower f1 when test training data set. So I finally choose (penalty=l2, C=15). It is probably because it is not a spark model by using l2, this model remains some small values not set to 0, which can get more accuracy while testing the data in other domains.