A Feature-based solution for Semantic Modeling
CS420 Coursework: Text Classification

Zhenjia Xu
xzj, 515030910637
Shanghai Jiao Tong University
xuzhenjia1997@gmail.com

April 19, 2017

Abstract

Text classification is a classic task of machine learning. There're various methods to solve this task, which are roughly divided into two categories, feature-based and end-to-end. I have tried most of these methods, and finally combined two feature-based methods of logistic regression and gradient boosting decision trees to solve this task. The overall performance of my model is not bad. The AUC in the final test data is 0.91508.

1 Introduction

1.1 Background

As aggregators, online news portals face great challenges in continuously selecting a pool of candidate articles to be shown to their users.
Typically, those candidate articles are recommended manually by platform editors from a much larger pool of articles aggregated from multiple sources. Such a hand-pick process is labor intensive and time-consuming. In this task, we study the editor article selection behavior and propose a learning by demonstration system to automatically select a subset of articles from the large pool.
This project aims to predict the probability of the given text piece belonging to the specific category (binary). Our goal is to train a classification model that provide estimated probability given the text.

1.2 Formulation

- Extracting useful features from the data, converting the test s to the feature vector $\boldsymbol{x}$.
- Selecting a model to make prediction $p_\theta(\boldsymbol{x})$: the predicted probability $Pr(\hat{y} = 1|\boldsymbol{x})$.
- Objective function:

$$J(\theta) = \sum_{\boldsymbol{x} \in \boldsymbol{X}_{train}} -y \log p_\theta - (1-y) \log(1-p_\theta) + ||\theta||_2^2$$

- The goal of the tast is to maximize the AUC of the prediction on the test data.

Table 1: Notations and descriptions

Notation	Description
$\boldsymbol{x}$	The feature vector of the article.
$\boldsymbol{X}_{train}$	The matrix of the training set.
$\boldsymbol{X}_{test}$	The matrix of the testing set.
y	The true label of user response.
$\boldsymbol{y}_{label}$	The true label vector of the train set.
$\boldsymbol{y}_{predict}$	The predicted label vector of the testing set.
p_θ	The predicted probability $Pr(\hat{y} = 1\|\boldsymbol{x})$.
θ	The parameters in the model.
$L(y, \boldsymbol{x}, p_\theta)$	The loss function of the objective.

2 Methodology

2.1 Feature extraction

Useing "Jieba" Chinese text segmentation to segment the text in the data.
Here're some features I use in the model:

- TextRank
  TextRank[2] is a graph-based ranking model for text processing.

- Tf-idf
  Tf-idf[9] is a numerical statistic that is intended to reflect how important a word is to a document in a collection or corpus

$$\begin{aligned} idf &= -\log P(t|d) \\ &= \log \frac{1}{P(t|d)} \\ &= \log \frac{N}{|\{d \in D \ : \ t \in d\}|} \end{aligned}$$

- Information gain
  Information gain[5] is a measure of the non-symmetric difference between two probability distributions P and Q.

$$\begin{aligned} IG(T) &= H(C) - H(C|T) \\ &= -\sum_{i=1}^{n} P(C_i) \log_2 P(C_i) \\ &+ P(T) \sum_{i=1}^{n} P(C_i|T) \log_2 P(C_i|T) + P(\bar{T}) \sum_{i=1}^{n} P(C_i|\bar{T}) \log_2 P(C_i|\bar{T}) \end{aligned}$$

- Chi-squared test
  A chi-squared test[3] can be used to attempt rejection of the null hypothesis that the

data are independent.

$$\begin{aligned} E_{11} &= N \times P(t) \times P(c) \\ &= N \times \frac{N_{11} + N_{10}}{N} \times \frac{N_{11} + N_{01}}{N} \end{aligned}$$

2.2 Logistic Regression

Logistic Regression[6] is a regression model where the dependent variable is categorical.

$$p_\theta(\boldsymbol{x}) = \frac{1}{1 + \theta^{\mathrm{T}}\boldsymbol{x}}$$

Cross entropy loss function:

$$L(y, \boldsymbol{x}, p_\theta) = -y \log p_\theta - (1 - y) \log(1 - p_\theta)$$

Gradient:

$$\frac{\partial L(y, \boldsymbol{x}, p_\theta)}{\partial \theta} = (\frac{1}{1 + \theta^{\mathrm{T}}\boldsymbol{x}} - y)\boldsymbol{x}$$

Optimization: mini-batch gradient descent

2.3 Gradient Boosting Decision Trees

GBDT[4] is a form of an ensemble of weak prediction models. Each weak model is a decision tree. The loss function and objective is the same as those in logistic regression.

2.4 Combination of models

I choose a simple but useful method to combine the prediction of logistic and gradient boosting decision trees together, that is, weighted average.

2.5 Some other methods

I also tried some other methods, such as Convolutional Neural Network[1], Support Vector Machine[8], Naive Bayes[7], but their performance are not good enough.

3 Experiments and Results

3.1 Data preparation

I devide the data into training set and evaluation set, the proportion of which is 80% and 20% respectively. The text segmentation I use is "Jieba", which can not only segment the test into words, but also mark the property of each word.

3.2 Logistic Regression

I use the Logistic Regression model in Scikit-learn, which is very convient to use. The most import hyperparameter in Logistic Regression is C(Inverse of regularization strength) and the optimal hyperparameter for different feature are different. The Figue 1 shows the relation between the hyperparameter C and the AUC for two different features.

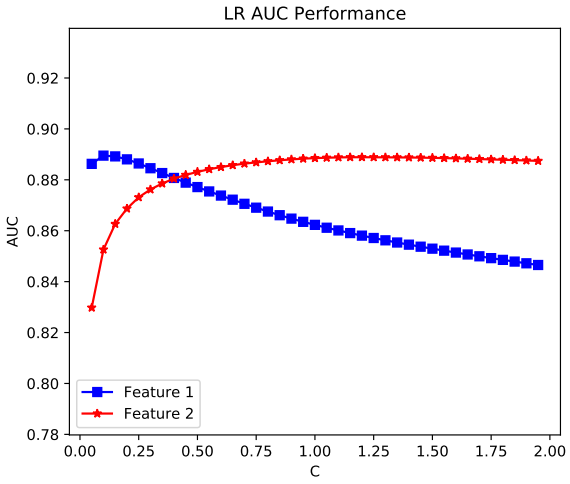

Figure 1: The performance of LR.

3.3 Gradient Boosting Decision Trees

I use XGBoost to implement the GBDT model. Two important hyperparameters in XGBoost is max_depth and n_estimeter, which represent the depth of each decision tree and the number of decision trees respectively. As is shown in Figure 2, these two hyperparameters together affect the performance of the model.

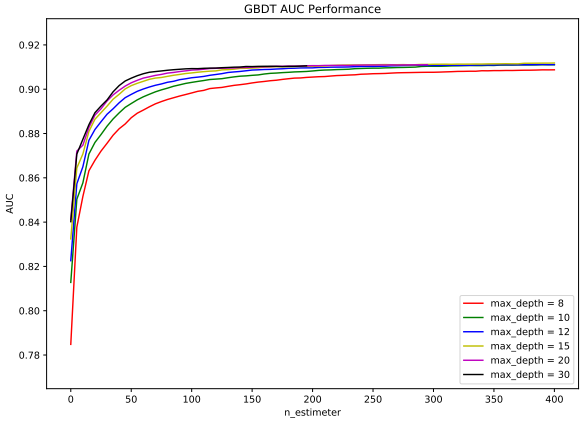

Figure 2: The performance of GBDT.

3.4 Combination

The AUC of Logistic Regression and Gradient Boosting Decision Trees are 0.899 and 0.912 resprectively. By using the simple weighted average method, the AUC can become 0.915. I also tried SVM and CNN, but these two models are running very slow, so I had to lower the size of the data and their performance is not good enough.

Table 2: AUC of different methods

Model	AUC
SVM	0.78
CNN	0.83
LR	0.899
$GBDT$	0.912
$Ensemble(LR\&GBDT)$	0.915

4 Discussion and Future works

Through this task, I have mastered some basic methods of data mining, including feature extraction, model selection, model combination, evaluation. I personally think that I don't do well in feature extraction because I only use some most common feature extration mathods, instead of extracting the most representative features. I have tried a large variety of models to make prediction, improving my ability to choose the most appropriate model. In terms of model combination, I thought of some good methods in the last few days, but I don't have much time to implement. As a result, I just use the simple weighted average method. I will try some other methods to extracting features and combine models when I have time. In summary, I have learned a lot from these course work and I will try my best to complete the next two course works.

References

[1] Y. Kim. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*, 2014.

[2] R. Mihalcea and P. Tarau. Textrank: Bringing order into texts. Association for Computational Linguistics, 2004.

[3] Wikipedia. Chi-squared test — wikipedia, the free encyclopedia, 2017. [Online; accessed 17-April-2017].

[4] Wikipedia. Gradient boosting — wikipedia, the free encyclopedia, 2017. [Online; accessed 18-April-2017].

[5] Wikipedia. Kullbackleibler divergence — wikipedia, the free encyclopedia, 2017. [Online; accessed 17-April-2017].

[6] Wikipedia. Logistic regression — wikipedia, the free encyclopedia, 2017. [Online; accessed 17-April-2017].

[7] Wikipedia. Naive bayes classifier — wikipedia, the free encyclopedia, 2017. [Online; accessed 18-April-2017].

[8] Wikipedia. Support vector machine — wikipedia, the free encyclopedia, 2017. [Online; accessed 18-April-2017].

[9] Wikipedia. Tfidf — wikipedia, the free encyclopedia, 2017. [Online; accessed 17-April-2017].

[10] W. Zhang, S. Yuan, and J. Wang. Optimal real-time bidding for display advertising. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1077–1086. ACM, 2014.