Web Page Type Classification

Xiangzhu Long, Xiao Han, Guan Wang
Carnegie Mellon University

Introduction

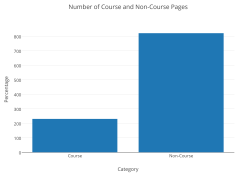
The World Wide Web has more than 4.72 billion pages by April, 2015 [1], most of them are unstructured data. Simply classifying them into different categories creates a lot of opportunities. For example, we can observe the possible relationships among people from the information extracted from the website. For recent decades, researchers have been exploring Machine Learning methods to create knowledge base and describe university people, courses, and research projects.[2]

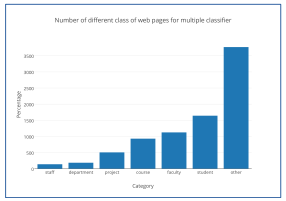
Our project is aim to classify web pages from five universities to four categories, professor, student, project or others. We explored Naive Bayes, Logistic Regression, Decision Tree and SVM on both binary and multi-classification on the content of website HTML files. And also, we use co-training for binary classification to improve result. Finally, based on the analysis of these methods, we implemented random forest, which consists of multiple decision trees to train this model.

Dataset

The original dataset contains webpages from 4 universities, labeled with whether they staff, department, project, course, faculty, student and other pages.[3] For binary classification, we combine the other categories except courses into non-course dataset.

The data set contains MIME information and raw html file, including all the HTML tags.





Methods

Naive Bayes(NB):

For the NB, we transferred each page into a vector X, where each element in the vector indicates the occurrences of a particular word. And Y is a vector of the labels of the pages.

For Training part, NB tries to maximize P(Y|X) = P(X|Y)P(Y), where P(X|Y) has more than 2^{n-1} parameters. But NB makes assumption that each dimension of features is independent of another dimension, which hugely reduce the computation to the complexity of O(n).

For Classification part, NB takes max{P(Y=class 1|X),P(Y=class 2|X)…P(Y=class 3|X)}.

Logistic Regression(LR):

LR is a way to deal with binary classification problem. It calculates the possibility of a word appear in a page with certain label. It makes no assumption at the dependency between two words, so the effect is supposed be better than NB.

We first treat a page as a bag of words. In training process, we calculate the different possibility of different word appearing with certain label. And for testing, we count all word that appears in the test set.

Decision Tree(DT):

Step 1: Calculate entropy of the target. Entropy is a measurement of the homogeneity of a sample.

Step 2: Get the information gain, which is the decrease in entropy.

Step 3: Build up the decision tree by choosing attributes with the largest information gain as the decision nodes and a branch with entropy of 0 as a leaf node.[1]

Support Vector Machine(SVM):

SVM is a technique which aims to maximize the margin between different labels of points. It is powerful not only because it maximize the margin, but because it can combine kernel transformation for better separation of data in different classes.

We used the same bag-of-words features as for algorithms above in SVM training process. The library provides us with several kernels ('linear', 'poly', 'rbf', 'sigmoid')

Co-training:

It's impossible to label every piece of websites on World Wide Web simply because of the large quantity. Thus, semi-supervised learning is an ideal way to use unlabeled data help shaping the distribution of the model. Co-training is one of such algorithm that aims to combine the the predictions of two independent feature sets.

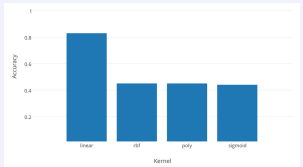
We modified piece of code named Co-Trade developed by a team in Nanjing University. The code used K-nearest neighbor criterion to decide the most confident unlabeled data to be labeled in each round of co-training.[4] The result is far beyond our expectation.

Random Forest(RF):

It's natural for us to think of using Random Forest since the performance on multi-classification using Decision Tree is pretty good. Random Forest is an algorithm that produces a model that consists of multiple decision trees. Because each decision tree "branch" is only trained on small piece of data and the model gets the votes to predict incoming data, it's general and robust to overfitting. We applied the algorithm on the multiple classification data, and it quietly improves the result we got from single Decision Tree.

Results

Comparison for SVM using different kernels:



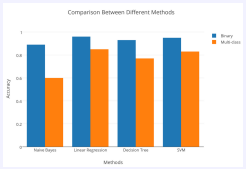
As shown in graph above, SVM with linear kernel outperforms other kernels.

Other kernels achieved such low performance may due to the fact that the original feature space for the dataset is too big and thus adding feature dimensions may cause the model to overfit.

Comparison between different methods:

70% of the data was used for training and 30% used for testing.

As shown above, Linear Regression achieved best performance on both binary and multi-classification.

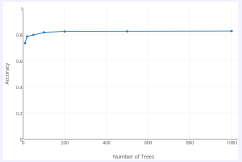


For Co-training:

We randomly chose 5 training data labeled with 'course' and 5 training data labeled with 'non-course', then we randomly chose 100 data samples to be test data. The rest of the data is unlabeled data which we used to improve the prediction in co-training. The algorithm achieved accuracy about 81% for the test set which is amazing given the number of labeled data we have! We compared it with decision tree using same number of training data, unlabeled data and testing data, decision tree only achieved better than random guessing but co-training increased the accuracy amazingly.

For Random Forest:

We also experimented with random forest. The result turned out to be 83%. It is better than decision tree. We tried different size of random forest and the accuracy is shown in following graph.



Conclusion

The linear regression achieved best performance among supervised learning algorithm we experimented here.

Compared to supervised learning algorithm, co-training achieved much better results given only a small set of labeled data.

For this specific question adding dimensions in data may cause models to overfit considering that the feature space is already large enough. But for other problems where features compared to number of data is not big, kernelization may be possible way to form better model

References

[1] Tom M. Mitchell, Machine Learning: "Chapter 3. Decision Tree Learning", ISBN: 0070428077, 2001

[2] Craven, Mark, et al. Learning to extract symbolic knowledge from the World Wide Web. No. CMU-CS-98-122. Carnegie-mellon univ pittsburgh pa school of computer Science, 1998.

[3] The 4 Universities Data Set, http://www.cs.cmu.edu/afs/cs.cmu.edu/project/theo-20/www/data/

[4] Min-Ling Zhang and Zhi-Hua Zhou, Senior Member, IEEE: COTRADE: Confident Co-Training with Data Editing