Predicting User Following Behavior on Tencent Weibo

Jinfeng Huang (jinfeng@stanford.edu); Hai Huang (hai928@stanford.edu); Zhaoyang Jin (zjin@stanford.edu)

Abstract Tencent Weibo, the Chinese Twitter-like service gains its popularity in recent years. Enormous commercial potential lies in the user behavior on the social media such as Tencent Weibo. In this project, different machine learning algorithms were applied to predict user following behavior on Tencent Weibo. The accuracy of prediction, true positive rate and true negative rates are estimated. The performance of algorithm in various training set size was evaluated. With experiments on five different algorithms, SVM with RBF kernel was determined to be the most effective method. As for feature analysis, user profile is shown to be the key feature subsets that make the biggest difference. Number of tweets stand out as the best indicator for the behavior of the user. With the findings in this project, Tencent Weibo was advised to further improve its recommendation practice with more emphasis on user information generally and the most active user group specifically.

1. Background introduction

Tencent Weibo is a Chinese Twitter-like service launched by Tencent, one of the biggest Internet companies in China. Since its launch, Tencent Weibo has become a major platform for sharing interests online. There are more than 200 million registered users on Tencent Weibo by far, generating over 40 million messages each day. This scale benefits the Tencent Weibo users but it can also flood users with huge volumes of information, hence putting them at risk of information overload.
Reducing the risk of information overload is a priority for improving the user experience and also presents opportunities for novel data mining solutions. Thus, capturing users' interests and serving them with potentially interesting items such as news, games, advertisements, products, is a crucial feature social networking websites like Tencent Weibo.

2. Project description

The goal of the project is to predict whether a user will follow an item that has been recommended to her. An item is a specific type of user on Tencent Weibo, which can be a person, an organization, or a group, that is selected and recommended to other users. Typically, celebrities, brands, or some well-known groups were selected to form the 'items set'. The raw data set contains two categories of data: user data and item data. User data includes user profile (ID, age, gender, personalized tags, # of tweeting times), following history, and users keywords with relative weights. Item data includes item ID, category and item keywords. We aim to use different classification algorithms to predict user behavior. By learning whether a specific user with certain background will recommend and follow a specific item, we can improve the personalized recommendation accuracy.

3. Data

3.1. Data Selection

With experimentation on different selection of features, we decide to keep the following seven features for our prediction model.

Category	Feature	Description
User Data	Gender	Integer value of 0, 1, or 2, for "unknown", "male", or "female", respectively.
	Year of birth	Integer selected by user when she registered.
	Number of tweets	Integer that represents the amount of tweets the user has posted.
Item Data	1st layer in item category	Items are organized in categories; each category belongs to another category, and all together forming a hierarchy. For example, 1st layer category "Science-and-technology", second layer category "Internet", third layer "Mobile" and forth layer "Company name"
	2nd layer in item category	
	3rd layer in item category	
	4th layer in item category	

Table 1: Training features for predicting whether a user will follow an item

We choose our features mostly based on performance on empirical trials. Besides, there are two more reasons which help us to make feature decisions: 1) Relevance. For example, data on follower-followee relationships history without detailed description does not provide us with useful information on a user's interest in a particular item. 2) Certain feature is too sparse to be useful. For example, there are millions of possible tags/keywords for a user. But one user may just have several even zero of them.

3.2. Data Statistics

We randomly select a considerable portion of data from what's provided by Tencent:

# of total data set from Tencent	70,000,000
# of total data we use with random selection	110,000
# of training data in our model	100,000
# of testing data in our model	10,000

Table 2: The sizes of training data and testing data

4. Methods

4.1. Algorithms

4.1.1. Support Vector Machine

A support vector machine constructs a hyper-plane or set of hyper-planes in a high or infinite dimensional space, which can be used for classification problem. A good separation is achieved by the hyper-plane that has the largest functional margin, since in general the larger the margin the lower the generalization error of the classifier.
We use two kernel functions in this report: radial basis function (RBF) kernel and linear kernel.
Radial basis function kernel is defined as: $\exp(-\gamma|x - x'|^2)$. γ must be greater than 0.
Linear kernel is defined as $\langle x, x' \rangle$.

4.1.2. Logistic Regression

Logistic regression is a linear model for classification rather than regression. In this model, the probabilities describing the possible outcomes are modeled using a logistic function. Our implementation fits a logistic regression with L2 regularization. As an optimization problem, binary class L2 penalized logistic regression minimizes the following cost function:

$$\min_{w,c} \frac{1}{2} w^T w + C \sum_{i=1}^{n} \log(\exp(-y_i(X_i^T w + c)) + 1).$$

4.1.3. Naive Bayes Classifier

Naive Bayes methods are a kind of supervised learning algorithms based on Bayes' theorem with a "naive" assumption. In this report, we implement the Gaussian Naive Bayes algorithm for classification. The likelihood of the features is assumed to be Gaussian:

$$P(x_i \mid y) = \frac{1}{\sqrt{2\pi\sigma_y^2}} \exp\left(-\frac{(x_i - \mu_y)^2}{2\sigma_y^2}\right)$$

The parameters σ_y and μ_y are estimated using maximum likelihood.

4.1.4. Stochastic gradient descent

Stochastic Gradient Descent (SGD) is a simple yet very efficient approach to discriminative learning of linear classifiers under convex loss functions. SGD has been successfully applied to large-scale and sparse machine learning problems often encountered in text classification and natural language processing. In this report, we use soft-margin linear Support Vector Machine (i.e. "hinge") as our loss function.

4.2. Implementation and Data Processing

We implemented the five algorithms using scikit-learn library version 0.15 in Python 2.7.3 in this report: SVM with RBF kernel, SVM with linear kernel, Logistic Regression, Naive Bayes Classifier and SGD with hinge loss function.
For comparing the accuracy performance of different algorithms, we randomly chose 100,000 lines of data from our raw data as our training set, and randomly chose another 10,000 lines of data as our test set. After we learned from training set and predicted labels for test set, we calculated three kinds of accuracy for the predicting result:
- Accuracy: defined as the number of successful predicting cases over the number of total test cases. We used the *average* value of accuracy of 10 times experiments.
- True positive rate: defined as in *worst* case, the number of successful predicting cases over the total number of positive cases (user follow item).
- True negative rate: defined as in *worst* case, the number of successful predicting cases over the total number of negative cases.

Note that true positive rate and true negative rate are defined under worst case, i.e. the lowest value of true positive accuracy / true negative accuracy in the 10 times experiments. We used these two numbers in order to measure the accuracy for unbalanced data, e.g., the positive cases exceed the negative cases, or vice versa. An algorithm in which true positive rate and true negative rate are both close to the total accuracy is regarded as a better one than algorithms in which these two rates are relatively low, or have large gap between them.
Then we tested the performance of different algorithms under different sizes of training sets in terms of the three accuracies defined above. Training sizes were set to 100, 200, 500, 1,000, 2,000, 5,000, 10,000, 20,000, 50,000 and 100,000 lines of data, and test case is set to 10,000 lines of data. These data were randomly chosen from the raw data. We plotted three accuracies versus

training size, and investigated how the increase of training size affects the performance of algorithm.

In order to analyze how each feature affects the performance in terms of accuracy, we choose the best algorithm from previous experiments, and run it 7 times. On each experiment, we left one feature out of our training features, and recorded the corresponding accuracy, true positive rate and true negative rate. The more the three accuracies decreases, the more important the leave-out feature is (Since we have already selected our features, the accuracies should not increase as one feature is left out). We listed the top three important features as well as the corresponding changes of accuracies.

5. Results

The following table summarizes the performance for all five methods we tried:

Method	Metrics	Perfor mance	Speed
SVM (RBF)	Accuracy	0.76	Medium (Around 30 minutes)
	True positive rate	0.78	
	True negative rate	0.74	
SVM (Linear)	Accuracy ratio	0.63	Slow (Around 2 hours)
	True positive rate	0.66	
	True negative rate	0.62	
Logistics	Accuracy	0.57	Fast (Less than 2 minutes)
	True positive rate	0.55	
	True negative rate	0.55	
Naïve Bayes (Gaussian)	Accuracy	0.55	Fast (Less than 1 minutes)
	True positive rate	0.86	
	True negative rate	0.2	
Stochastic gradient descent (Hinge)	Accuracy	0.5	Fast (Less than 1 minutes)
	True positive rate	0.01	
	True negative rate	0.01	

Table 3: Algorithms' accuracy, true positive rate, true negative rate and implementation speed for 100,000 training examples and 10,000 testing examples

For a fixed amount of 100,000 training data, we notice that SVM with RBF kernel delivers the best result for two metrics: accuracy (0.76) and true negative rate (0.74). Its true positive rate (0.78) is high too, but only ranks second. Naive Bayes gives the highest true positive rate(0.86). However, this is not a good prediction algorithm because such performance comes with the price that its true negative rate is low, meaning the Naive Bayes algorithm is strongly biased towards positive results.

A second significant result from our experiment is that stochastic gradient descent is not a solid method for this problem, with the worst case rendering only 1% for true positive rate and true negative rate! Further more, its average accuracy is barely 50%, no better than any random prediction. In high dimensional space, stochastic gradient descent, as linear classifier, does not fit the problem.

Another interesting pattern worth noticing is that the true positive rate is generally higher than true negative rate, and the accuracy, resulted from the combining effect from true positive rate and true negative rate, stays in between. We deem this pattern as desirable. As in real life, Tencent Weibo will definitely be more interested in "getting things right", namely how they could get people to follow an item. We will discuss more of this phenomenon in the following section.

Finally, a better algorithm in terms of accuracy may suffer in efficiency. SVM with RBF kernel is the high performing algorithm but it took thirty minutes to run. Other algorithms do not perform as well could be very fast with run time less than a minute. There is trade off between accuracy and efficiency.

5.1. Accuracy: Effects of increased size of training data

We also want to explore, for each method, how the size of training data will affect the performance of the algorithm. To make results comparable, we keep the 10:1 ratio between training data and test data across all experiments. The following chart summarizes the result:

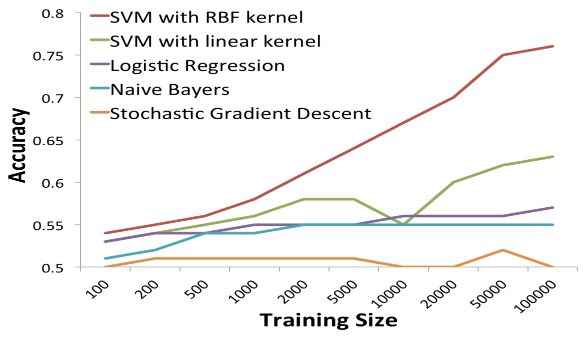

Figure 1: Algorithms' accuracy over different training size

Except for stochastic gradient descent, all other algorithms deliver significantly better performance with more training data. For best algorithm, namely SVM with RBF kernel, the performance boost from

100 training data to 100,000 data can be a dramatic, from 54% to 76%.

From the chart above, we could also draw an important conclusion that for this particular application, SVM algorithm will deliver significantly better performance with a RBF kernel than with a linear kernel.

5.2. True positive rate and true negative rate: Effects of increased size of training data

The following two charts summarize each algorithm's performance measured in true positive rate and true negative rate respectively:

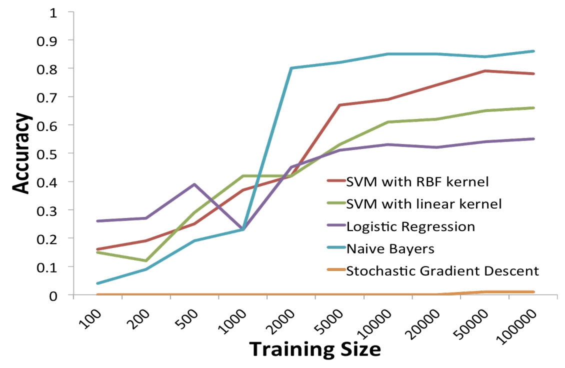

Figure 2: Algorithms' true positive rate over different training size

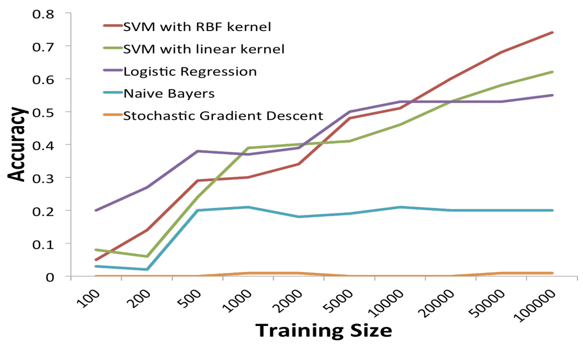

Figure 3: Algorithms' negative positive rate over different training size

We could conclude from the charts above that for all five algorithms, true positive rate beats true negative rate with the same training data size almost under every situation. It means that with the seven features, these algorithms tend to successfully predict positive results (i.e. follow or recommend an item), which is preferable in applications. The gap between true positive rate and true negative rate is usually not significant except in Naïve Bayes Classifier.

Another significant trend as training data increases is that true positive rate and true negative rate tends to converge to overall accuracy. Using SVM with RBF kernel as an illustrating example (other algorithms also generate similar trends except Naive Bayes):

Data size	100	500	1000	5000	10000	50000	100000
Accuracy	0.54	0.56	0.58	0.64	0.67	0.75	0.76
True positive	0.16	0.25	0.37	0.67	0.69	0.79	0.78
True negative	0.05	0.29	0.3	0.48	0.51	0.68	0.74

Table 4: Accuracy trends with increasing training set in SVM (RBF kernel)

This shows that all algorithms' bias for positive results come to diminish with the help of big training data size.

5.3. Feature Analysis

We also try to explore which feature contributes most to our prediction so that we could understand the Tencent Weibo user behavior in depth. The following chart summarizes the top 3 features measured in how much accuracy will be compromised if we remove the feature in the prediction model:

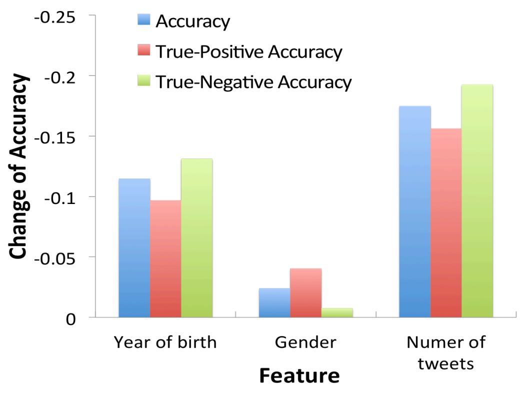

Figure 4: Top 3 critical features for model performance

There are two important insights from the above chart. Firstly, number of tweets a user had sent is the best indicator of whether she would follow a recommended item. Given this important information, Tencent Weibo could consider concentrating more recommendation efforts to the most active user group.

Secondly, all top 3 features (number of tweets, year of birth, gender) concern the user profile rather than item characteristics. This fact illustrates that to determine whether a user will follow a particular item, the information on the user herself is more valuable than the information on the item.

6. Conclusion

With experiments on five different algorithms, we are able to determine the best one: SVM with RBF kernel. This algorithm delivers 76% accuracy 100,000 training data and a test set of 10,000 data. The positive true rate (78%) and negative true rate (74%) is also reasonably similar, showing no bias between positive and negative results.

We are also able to demonstrate interesting trends with increased data size. Except for SGD, accuracy for all other four algorithms experience considerable improvement in performance as training data size increases. The true positive rate and true negative rate also shows similar patterns. Furthermore, although true positive rate is significantly higher than true negative rate with smaller training set, such bias is diminished as training set gets larger and both metrics converge to overall accuracy with the 100,000 training set size.

As for feature analysis, we show that user profile is the key feature subsets that make the biggest difference. Among three user profile features, number of tweets stand out as the best indicator of whether a user will follow a recommended item. Tencent Weibo is advised to further improve its recommendation practice with more emphasis on user information generally and the most active user group specifically.

7. Further work

7.1. Use User Following History

Constrained by the data source released by Tencent, some of the findings in our project seem to be trivial. More interesting results could be found if we have the knowledge of what the user's interests are. For example, if we could get keywords and tags of items a user follow in the past, and thus match with the user's own keywords and tags, we could hopefully raise the model performance to the next level.

7.2. Use Social Graph to Determine Following Patterns

In our model, we did not explore the relationship between the following behavior of a user's friend and the behavior of the user herself. An interesting hypothesis that could be tested is that one's interest is similar to one's friends'.

7.3. Use the Popularity of Each Item

We did not differentiate the item's own popularity in our model. This is a high potential improvement possibility for us because a popular item is intrinsically more likely to get following when presented to a user. Taking into account this effect, we could change the current situation where item characteristics do not contribute enough to the prediction.

Reference

1. Data source: http://www.kddcup2012.org/
2. Scikit-learn library documentation: http://scikit-learn.org/stable/documentation.html
3. CS 229 Lecture Notes, Stanford University, 2014
4. C. M. Bishop, Pattern Recognition And Machine Learning, Springer, 2006
5. K. P. Murphy, Machine Learning: A Probabilistic Perspective, the MIT press, 2012
6. T. Hastie, R. Tibshirani and J. Friedman, The Elements of Statistical Learning: Data Mining, Inference, and Prediction (2nd edition), Springer, 2008
7. C. J. C. Burges, A Tutorial on Support Vector Machines for Pattern Recognition, Data Mining and Knowledge Discovery, 1998
8. J. Shawe-Taylor and N. Cristianini, Kernel Methods for Pattern Analysis, Cambridge University Press, 2004
9. What is an RBF kernel: https://charlesmartin14.wordpress.com/2012/02/06/kernels_part_1/
10. Kernel Functions for Machine Learning Applications: http://crsouza.blogspot.com/2010/03/kernel-functions-for-machine-learning.html
11. J. M, Hilbe, Logistic Regression Models, Chapman & Hall Press, 2008
12. Andrew Y. Ng, Feature selection, L1 vs. L2 regularization, and rotational invariance, ICML Conference, 2004
13. S. Russell and P. Norvig, Peter, Artificial Intelligence: A Modern Approach (2nd edition), Prentice Hall, 2003
14. L. Bottou, Stochastic Gradient Descent Tricks, Microsoft Research, 2012