Byte Cup Challenge 2016-Recommender System for Toutiao

Ravishankar Sivaraman, Mahesh Pottippala Subrahmanya, Adarsha Desai
University Of Southern California
Los Angeles
`rsivaram@usc.edu,pottippa@usc.edu,adarshad@usc.edu`

Abstract

In this project we try to solve the problem for Byte cup challenge 2016. The data set contains the user information and data related to the questions asked by the user. Also the data is available about the experts who answer the questions. Given a question and an expert pair, the problem is to predict the probability of that expert answering the question. We try to solve this problem using several machine learning algorithms and at the end we will pick the best algorithm based on its performance.

1 Introduction

Toutiao Q and A is an up-and-coming mobile social platform, built upon 530 million Toutiao users and precise recommendation algorithm, promoting short-form content creation and interaction on mobile devices in the format of Q and A. Toutiao Q and A has tens of thousands users answering on a daily basis, while user-created answers are viewed tens of millions times a day. An important question is how to match questions with interested users' expertise. If the matching strategy is not accurate enough, to ensure that questions get enough top quality answers, the system will have to send out invitations to more expert users who may be disturbed. Each data record includes expert tags, question data and question distribution data. Given certain questions, we need to forecast which experts are more likely to answer which questions. Specifically, given each question and each expert, we need to calculate the probability of that expert answering the question. The problem can be solved using the conventional classification algorithms such as SVM, Random forest classifier, Logistic Regression etc. Another approach is to solve the problem using recommender systems such as collaborative filtering. we will have to asses these algorithms based on their performance on the validation dataset. The best performing algorithm can be selected as the final solution.

2 Data Pre-processing

2.1 Data representation

The data is given in two parts 1.Question data: This data is about the questions asked by the user. It has the following format.

Question ID	Question Tag	Word ID sequence	Character ID sequence	Number of Up votes	Number of Answers	Number of top quality answers
c1c0075239841777d5b01c40b38135d2	0	0/1/2/3/4/5/6/7	0/1/2/3/4/5/6/5/7/8/9/3/10	103	6	5

Question ID: this is encrypted question ID of the question asked by users.
Question tag: This is like a category such as sports/health etc Word ID sequence: These are the keywords which are extracted from the question.
Character ID sequence: These are the characters which are extracted from the questions.
Number of up votes: This attribute shows that how many people find the question-answer as relevant one.
Number of answers: Again this shows the popularity of the question.
Number of top quality answers: This attribute shows the expertise about that field or popularity of the question.

2. Expert data: This is the data related to all expert users. Below is the format of the data.

User ID	Expert Tag	Word ID sequence	Character ID sequence
4588a1df2461674252ff01c63b59171a	2/3/4/5/6	54/11880/113/13231/13232/113/8864/444/7404	92/93/160/160/183/1022/1022/183/732/732/183/1449/2193/11/663/188

User ID: encrypted user ID of the user/expert.
Expert tag: This shows various fields in which the current user is an expert.
Word ID sequence: These are the keywords which are frequently used by the expert.
Character ID sequence: These are the characters which are frequently used by the expert.
Finally, we are given with the question distribution data which says whether the user has answered the question or not.

2.2 Pre-processing

Each attribute such as category number etc can be considered as a dimension of each data point. Also we can see that few attributes such as word ID and character ID are given as a sequence of numbers. They should be split so that they can be used as features of the data point. Example: below is the example for splitting the strings and making them as features.

	1	2	3	4	5	6	7	8
2/3/4/5/6	0	1	1	1	1	1	0	0
1/2/4/7	1	1	0	1	0	0	1	0

As we can see that the data is distributed in two parts we need to combine them in order to make a matrix of data points for training. We have to get the question ID and corresponding information from the question info file such as word ID sequence and also the corresponding information from the user info file such as expert tag. After getting the information with respect to each pair of question ID and user ID we will stack them and form a matrix of NxM with N data points and M dimensions. This is our data matrix. All the processing will be done on this matrix.
Training data: we have 245752 data points. These are the QID (question ID) and User ID pairs with the target value saying if the user will answer the question or not.

Testing data: we have 30466 data points as validation data set. We need to give the probability of the given User answering the given question.

3 Approaches:

3.1 Classifiers:

The problem can be seen as a classification problem. Where the class 0 represents the user does not answer the question and class 1 represents the user answers the question. By using the data we had pre-processed and using the labels 0/1 from the training data we can apply the typical classification algorithms and get the probability score from the algorithm of that data point belonging to class 1. This probability is the probability of that user answering that question.
Below are the algorithms that support the probability score from scikit learn:

3.1.1 Logistic Regression:

This is a classifier using sigmoid function for the decision boundary. So we get the probability scores inherently from this classifier. L2 regularization is used to prevent the over fitting. This is tuned based on cross validation.

3.1.2 SVM:

This is really good classifier considering that there are only two classes. This uses the support vectors and kernelization trick to classify the data. Kernel parameters, slack penalty are the hyper parameters and they are tuned based on cross validation.

3.1.3 Random forest:

This is just a collection of decision trees. Each decision tree is constructed using a fraction of features. Thus each tree will be constructed from mutually exclusive features and they will be collectively exhaustive to cover all the features of the data. During classification, the data point is classified separately by all the trees. After this considering the voting from the trees the final class will be decided. Number of trees and depth of trees are hyper parameters they are tuned based on cross validation. Random forest also gives the probability scores for the classes.

3.2 Recommender Systems:

In the previous section we have discussed about the typical Machine learning classifiers. We can use the recommender systems to guess the probability of the user answering the question. This method appears more sensible in terms of the model that is being used to solve the problem. This method takes in to account of user -item relation as well as user-user co-relation in to account and tries to fit the current data to the model. Once the model is learnt during the classification phase, the probability for a specific user could be directly obtained from the matrix. It's based on matrix factorization-SVD based principle. We are using graphlab library to use these recommender systems. The graphlab library supports many types of recommender system.

3.3 Different types of recommender systems:

3.3.1 Matrix factorization:

The basic idea behind matrix factorization is that given a non negative matrix V, we find non-negative matrix factors W and H such that V=WxH. This method can be used to mode/discover the latent features underlying the interactions between two different kinds of entities. If we can model these latent features, we will be able to predict the rating that the user would give the item, because the combined features of the user and item should be consistent. Or the features of item should match the features of the user for a specific rating.

3.3.2 Item content based filtering:

In this method, item -item similarity/interaction is taken in to consideration to build the model. Based on this a profile is built for the users who like one kind of item. The recommendations are made as per the average similarity of a candidate item to all the items in a user's set of rated items.

3.3.3 Hybrid methods:

The usage of multiple algorithms to predict the output can also be used. The method such as: a.Combining the output of SVM and Matrix Factorization. b.Combining the output of Logistic regression with Matrix Factorization. The mixture averages could also be tuned such as to considering both algorithms in same weightage or giving more weightage to one of them.

4 References

[1]Toine Bogers, Antal van den Bosch, "Collaborative and Content-based Filtering for Item Recommendation on Social Bookmarking Websites", Workshop on Recommender Systems and the Social Web, 2009.
[2]Yuchen Zhang,Amr Ahmed, Vanja Josifovski, Alexander Smola, "Taxonomy Discovery for Personalized Recommendation", The Seventh International Conference on Web Search and Data Mining, 2014.
[3]https://turi.com/
[4]http://scikit-learn.org/stable/

5 Results:

Algorithm	Library	Accuracy on validation set	Note
Random forest	Scikit learn	0.247189	depth of tree and number of trees were hyper parameters which were tuned
Random forest	Scikit learn	0.221184	
Random forest	Scikit learn	0.246781	
Ranking factorization recommender	Graphlab	0.259723	Use Ranking factorization recommender system in Graphlab
Matrix factorization (Content based filtering)	Graphlab	0.458673	Use factorization recommender in Graphlab also tuning for number of iterations for SGD
Matrix factorization +SVM	Graphlab	0.448748	Use average of output of factorization recommender and SVM in Graphlab
Matrix factorization+Logistic Regression	Graphlab	0.452045	Use average of output of factorization recommender and logistic regression in Graphlab
Matrix factorization +SVM+Logistic Regression	Graphlab	0.433164	Use average of output of factorization recommender, Logistic regression and SVM in Graphlab

6 Conclusion:

The Matrix factorization gives the best result for validation data set. The recommender system of graphlab uses the data of the user info and question info both to fit the model. Other probable approaches to solve: This data could be modeled as a multiclass -multilabel classification problem. Each class being the expert and data point being the question. By this we can get the probability of question belonging to each class/expert. Thus the maximum probability gives the best class and proper thresholding gives the probability of answering the question.