CSCI 567 Fall 2016 - Mini Project

Deep Sidhpura, Rajvi Mehta, Neelam Gehlot
(Team RND)

December 11, 2016

Abstract

The following report highlights the various models and approaches implemented by the team "RND" in the 2016 Byte Cup Machine Learning Competition. The task of the competition was to analyze the data set given by Toutiao Q and A to train a model in order to predict if an expert will be able to answer the particular question asked on the Toutiao forum.

1 Introduction

In this competition of Machine Learning, the challenge was to match community raised questions with domain experts. Toutiao is a Q and A platform upcoming mobile social platform. It is built upon 530 million Toutiao users and 300 thousand professional freelance writers, along with precise recommendation algorithm, promoting short-form content creation. The task was to match information with the right people, finding the best respondent to the questions, and the best readers to the answers. Data provided by Toutiao Q and A was used to conduct analysis on specific questions. It was necessary that matching strategy be accurate to ensure that questions get enough top quality answers, otherwise the system will have to send out invitations to more expert users who may be disturbed. Each data record included expert tags, question data and question distribution data. Given certain questions, participants need to forecast which experts are more likely to answer which questions. The outcome was to calculate the probability of that expert answering the question.

2 Data

The data consists of information about users(experts), questions and binary label if particular user answered given question(Question distribution).

The training dataset consist of 28763 users information, 8095 questions information and 245752 observations about expert answering or ignoring given question. Validation dataset consist of 30466 enteries of users and questions ID.

2.1 Description of Data

- The users(experts) information includes
    1. Anonymized user ID
    2. IDs of topics in which user is an expert separated by slash
    3. IDs of each word in user's description separated by slash
    4. IDs of each character in user's description separated by slash
- The questions information includes
    1. Anonymized question ID
    2. ID of the topic to which question belongs
    3. IDs of each word in question description separated by slash
    4. IDs of each character in the question description separated by slash
    5. Total number of upvotes to all the answer to the question which signifies the popularity if the question
    6. Total number of answers given for the question
    7. Number of top answers for the question

Expected label for the validation dataset was the probability of particular expert answering the given question.

2.2 Data Preprocessing

- One hot Encoding of User Information: o First step was to extract the unique user tags, unique characters and unique word of the entire user(experts) information o Create column name for all the unique tags, characters and words o Finally, discretize it. For instance, if unique expert tags obtained were: 1,2,3,4 and user is an expert in 1, the column 1 would be marked 1 and rest would be marked 0 - One hot Encoding of Question Information: o First step was to extract the unique characters and unique word of the entire question information o Create column name for all the unique characters and words o Finally, discretize it in the similar fashion. - Creation of final train Data set: o The training set was obtained by joining the user and question table obtained from above. o The inner join was based on primary key as Question ID and User ID based on question distribution information. o As a result of join operation, there were some columns containing only 0, such columns were dropped from the data set. o Two kind of data set was created, one containing only characters as the features, other containing both word and character as the feature.

2.3 Feature Engineering

As mentioned in the Data Preprocessing section above, 2 kinds of data sets were used: one which involved only the character features and the other which involve both one hot encoded word and character features. Since the character features involved around 6000 columns, several models were trained on the entire data set consisting of all the 6000 features. However including the word and character features both required some form of dimensionality reduction technique. To reduce the dimensions, a technique known as Truncated Singular Valued Decomposition which is also known as Latent Semantic Analysis was used. This transformer performs linear dimensionality reduction by means of truncated singular value decomposition (SVD). Contrary to PCA, this estimator does not center the data before computing the singular value decomposition. This means it can work with sparse matrices efficiently, as obtained by one-hot encoding our data set. Again several models were tried on this data set as well and a better accuracy was obtained. The description of the models used along with the data sets used to obtain them is described in the next section.

2.4 Evaluation Metric

The metric used for evaluating the accuracy of the models on the validation and the test set was .the mean of NDCG@5 and NDCG@10. Given a certain question, experts were ranked based on the forecasted probability, and NDCG@5 and NDCG@10 of ranking results were evaluated. The final evaluation formula used was: NDCG@5 * 0.5 + NDCG@10 * 0.5.

3 Models

This section describes the models that were developed using the data set prepared as described in the above section. It also mentions the accuracy achieved on the validation set and the test set in terms of the evaluation metric described previously.

3.1 Random Forests

The first model that was tried on the data set was a Random Forest classifier.A random forest is a meta estimator that fits a number of decision tree classifiers on various sub-samples of the data set and use averaging to improve the predictive accuracy and control over-fitting. The sub-sample size is always the same as the original input sample size but the samples are drawn with replacement if bootstrap is used.

The Random Forest algorithm when applied on the data set that involves only character features obtained the following accuracies:

- Random Forest with 200 trees and sqrt(features) per tree= 0.45799

- Random Forest with 500 trees sqrt(features) per tree = 0.46073
- Random Forest with 500 trees and 200 features per tree = 0.4546

Increasing the trees slightly improved the accuracy. Increasing the number of features per tree led to a slight over fit and gave a lesser accuracy.

The Random Forest algorithm applied on the data set with 6000 truncated SVD features performed as below:

- Random Forest with 500 trees and sqrt(features) per tree = 0.46955
- Random Forest with 1000 trees and sqrt(features) per tree = 0.47447

  Thus the Truncated SVD feature engineering technique gave a better result.

3.2 Extreme Gradient Boosting (XGB)

XGBoost is an extremely fast, scalable implementation of Gradient Boosting Machines, which can be used for both classification and regression tasks.Due to its performance in various machine learning competitions, XGBoost was a favorable choice for the model. The accuracy obtained using XGBoost is as follows:

- XGBoost with 3000 trees, learning rate of 0.03, min child weight of 7 and a maximum depth of 10 = 0.4448
- XGBoost with 6000 trees and the same parameters as above = 0.4634

Applying XGBoost model on the Truncated SVD data set obtained an accuracy as follows:

- XGBoost with 100 trees, learning rate of 0.1 min child weight of 7 and a maximum depth of 10 = 0.4514

Training the XGBoost on Truncated SVD features took a lot of time and thus not enough parameter tuning was experimented. A much better accuracy was expected of this model, however the number of hyper parameters to fine-tune required time and resources, not enough of which was available at our disposal.

3.3 Neural Networks

Since the data set had around 250,000 training samples, which was a huge data set, it was ideal to implement Deep Learning. A Deep Learning Network with 4 hidden layers with the specification of [input, 800, 500, 300, 200, output] was trained using Stochastic Gradient Descent and Momentum. Some amount of l2 regularization was also utilized. The accuracy obtained was 0.4356.

Once again as was the case with XGBoost, Neural Networks took a long time to train and the number of hyper parameters used in Neural Networks required a lot of fine tuning that required a lot of time, not available with us at that time.

Neural Networks and XGBoost models were expected to outperform any other model as seen in several machine learning competitions, however fine tuning them was the difficult and time consuming task.

3.4 Extra Trees Classifier

It is a meta estimator that fits a number of randomized decision trees (extra-trees) on various sub-samples of the data set and use averaging to improve the predictive accuracy and control over-fitting.

The Extra Trees Classifier was trained on the data set prepared by Truncated SVD obtained the following accuracy:

- Extra Trees Classifier with 800 trees and min sample leaf of 12 = 0.48833
- Extra Trees Classifier with 700 trees and min sample leaf of 12 = 0.4877858

As seen from the results, the Extra Trees classifier gave the best results on the validation set.

4 Implementation

All the algorithms mentioned above were implemented in Python using many libraries. For Random Forests, Extra Trees Classifier and Truncated SVD, Sci-Kit Learn library was utilized. XGBoost Python Library was utilized for training the Gradient Boosted Machines. The Keras library was utilized for Neural Networks. All of these algorithms are implemented as separate Python Files that read the data and output a csv prediction file. Before running the algorithm files, the preprocessing.py file needs to be implemented and the user data and the question data needs to be inner joined to generate the final data.

5 Results

The Extra Trees Classifier algorithm gave the best accuracy on the validation set with the NDCG score of 0.48833. The ranking on the leader board was 253. The score on the final test set was 0.47132 with a ranking of 75. The table 1 summarizes the results of all the algorithms and models tried.

References

1. Abhishek Thakur Kaggle Blog Approaching (Almost) Any Machine Learning Competition
2. Machine Learning Libraries: Sci-kit learn, XGBoost, Keras etc.

Table 1: Summary of Results

Algorithm	Parameters and Features	Leaderboard Score
Random Forest	300 Trees, Character Features only,sqrt(features) per tree	0.45799
Random Forest	500 Trees, Character Features only,sqrt(features) per tree	0.46073
Random Forest	500 Trees, Character Features only, 200 features per tree	0.4546
Random Forest	500 Trees, Truncated SVD features,sqrt(features) per tree	0.46955
Random Forest	1000 Trees, Truncated SVD features,sqrt(features) per tree	0.47447
XGBoost	3000 Trees, min_child_weight=7,eta=0.03,max_depth=10	0.4448
XGBoost	6000 Trees,min_child_weight=7,eta=0.03,max_depth=10	0.4634
XGBoost	100 Trees,min_child_weight=7,eta=0.1,max_depth=10	0.4514
Neural Networks	[input, 800, 500, 300, 200, output], Character Features only	0.4356
Extra Trees Classifier	800 Trees, min_sample_leaf=12,Truncated SVD features	0.48833
Extra Trees Classifier	700 Trees, min_sample_leaf=12, Truncated SVD features	0.4877858