**Predict which Users one user might follow in Tencent Weibo**

**Ravi Teja Yaramaneni, Satya Abhignya Maddali**

**Introduction:**

There has been a rapid growth in the amount of data available. The primary concern of today is to store, process, analyze, interpret, consume the massive amounts of information each organization produces. There has been a 50-fold growth in the amount of data available. In the past 3 years there has been an exponential growth of data only in the social media.

Tencent Weibo is one of the largest micro-blogging websites in China where people increase their social network and share their interests. Users can share photos, videos and texts. Currently it has 200 million registered users who generate around 40 million messages per day. The users get various suggestions and this leads to information overload. Thus, to improve the user experience, the information provided to them must be restricted. The most important feature of any social networking platform is to give the user a good set of information based on their posts or interests.

**Problem Description:**

The scope of our project is to predict whether a user will follow an item recommended to them. The recommendation can be any person, organization or groups.

**Related Work:**

These days every website tries to recommend a product based on the explicit and implicit user generated data regarding a product. Ex: Netflix, Amazon etc.

**Dataset description:**

In the website Tencent Weibo, an item is a user, an organization or a group. These items are grouped together and are known as categories. These items were selected and recommended to other users based on their history. The dataset consists of user’s preferences and the recommendations.

The training dataset is *rec\_log\_train.txt* and the testing dataset is *rec\_log\_test.txt.* Each file consists of the following fields:

* **UserId:** It is a random and anonymous number that is assigned to a user.
* **ItemId:** It is a random and anonymous number that is assigned to an item.
* **Result:** If the recommendation is accepted by the user it is a 1 or else it is -1. The training data contains the true values of Result

**(UserId)\t(ItemId)\t(Result)\t(Unix-timestamp)**

**User\_profile.txt** - This contains the information of each user like the date of birth which is selected by the user at the time of registration, gender is ‘0’,’1’ and ‘2’ based on unknown, male and female, number of tweets and tag ids are the number of interests selected by users.

**(UserId)\t(Year-of-birth)\t(Gender)\t(Number-of-tweet)\t(Tag-Ids)**

**Item.txt***-* These are sets that contain both the keyword and the category. The item-category is a string and are ordered in top-down fashion, the item-keyword are keywords extracted from the user profile.

**(ItemId)\t(Item-Category)\t(Item-Keyword)**

**User\_action.txt** - This contains the number of times a user has used “@” to another user in the recent days. UserId is the user who is using “@” , Action-Destination-UserId is the user who was addressed to, Number-of-at-action is the number of times they used “at”, Number-of-retweet is the number of retweets between the two user and Number-of-comment is the number of comments between the two.

**(UserId)\t(Action-Destination-UserId)\t(Number-of-at-action)\t(Number-of-retweet )\t(Number-of-comment)**

**User\_sns.txt** - This file contains each users follow history.

**(Follower-userid)\t(Followee-userid)**

**User\_key\_word.txt** *-* These are the words the user is interested in. They are extracted from the user’s comments/ posts and tweets.

**(UserId)\t(Keywords)**

**Proposed solution and Methods:**

We want to test the data using two models ALS and Gradient boosted technique.

In the ALS (collaborative filtering) method we require only the *rec\_log\_train.txt,* as it requires only the UserId , ItemId , rating as input to predict their interest in a product. ALS has the capacity to generate its own feature matrices for the user and products based on the user item interactions.

In Gradient boosted technique(ML) method we require *rec\_log\_train.txt*, *user\_key\_word.txt, item.txt.* We are doing this under the assumption that only the keyword correlation between items and users will be sufficient to recommend an item to a user. To estimate the model, we require all the features of the products and users to estimate the behavior and predict the likelihood of a user accepting a product.

**Pre-processing techniques:**

The **entire dataset** size i.e. the rec\_log\_train.txt has **73209277** lines in it with user data. Out of this huge data the user's **accepted** recommendation dataset size is **5253828**, which is very small compared to the entire dataset. Thus, running algorithms on such type of data would making the prediction model converge to 0. So, we included all the positive recommendations and sampled data of the negative recommendations data which is 13594730.

The total of both positive and negative recommendations is **18848558.**

Due to this large amount of data, majority of the time was spent on data preprocessing. The cluster was not able to process this huge amounts of data and user features. Thus, we took only a fraction of the entire data.

For the gradient boosted tree, we wanted to generate recommendation based on the item key words (in the file **Item.txt**) and the key words used by the user (in the file **User\_key\_word.txt**).

Initially, we decided to generate key word vectors of each user. The key word feature vector space for both user and items were very sparse. The size of each vector was 713747 which was very large and this became a major challenge. Conventional methods like PCA could only solve vectors of size 65535. To overcome this challenge, we used Word2Vec. This has an estimator which trains a sequence of words into a Word2Vec Model. Each word will have a unique fixed size vector. The Word2Vec Model then converts all the documents to a vector which can be used to predict features.

In our project, we considered each feature vector of a user as a document and sorted all the features in ascending order. The Word2vec was applied to the document which generated unique vectors for each document based on dictionary and priority of the word. By using this method, we could keep all the vital information for the user and the items with respect to others. Thus, using this helped us maintain the correlation between users and items and with other users and items.

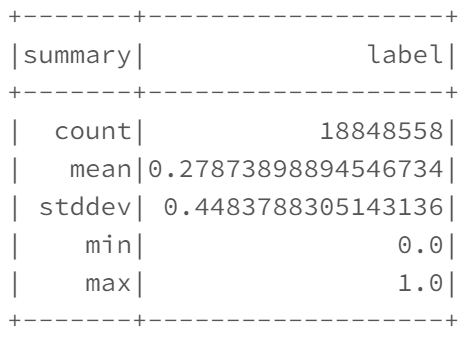
In the ALS model, we just need the user/item/recommendation vector to train the model. There wasn’t much feature extraction or processing required. The input to the ALS was the subset of the main data generated previously in the above steps.

**Implementation:**

* **Gradient Boosted tree:** In this method, we used the keyword of items and users a feature for training the model. The features we sparse and running PCA on them is not advisable and the spark PCA library cannot handle vectors greater than 65535. For this each user feature and item feature is converted in a vector of dimension 5 using word2vec. These features and the output is fed into the gradient boosted tree model. The data is divided into training and testing set in the ration of 80:20. Once the model completed training on the train data. Test data is used to check the accuracy for the model.
* **ALS method:** In this model, we used the userid, itemId and recommendation. We considered the recommendation as the rating and trained the model. The data set is divided into training and testing data in the ratio of 80:20. The output rating of the test data is in the range of -1 to 1. For recommendation system values between -1 and 1 are not allowed so, we made all values in [0,1] to 1 and all the values between [-1,0) to -1.

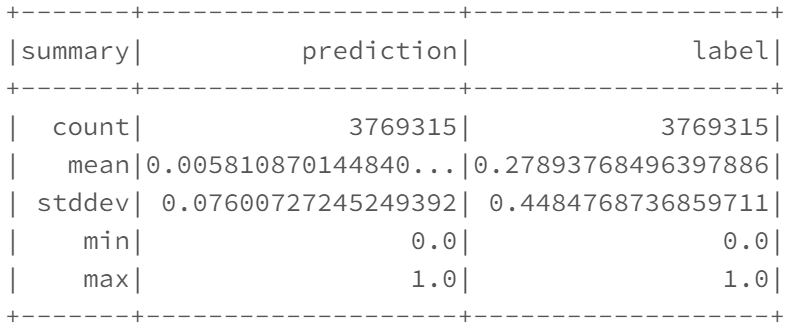
**Results and Analysis:**

**Statistics of Input data for GBT:**



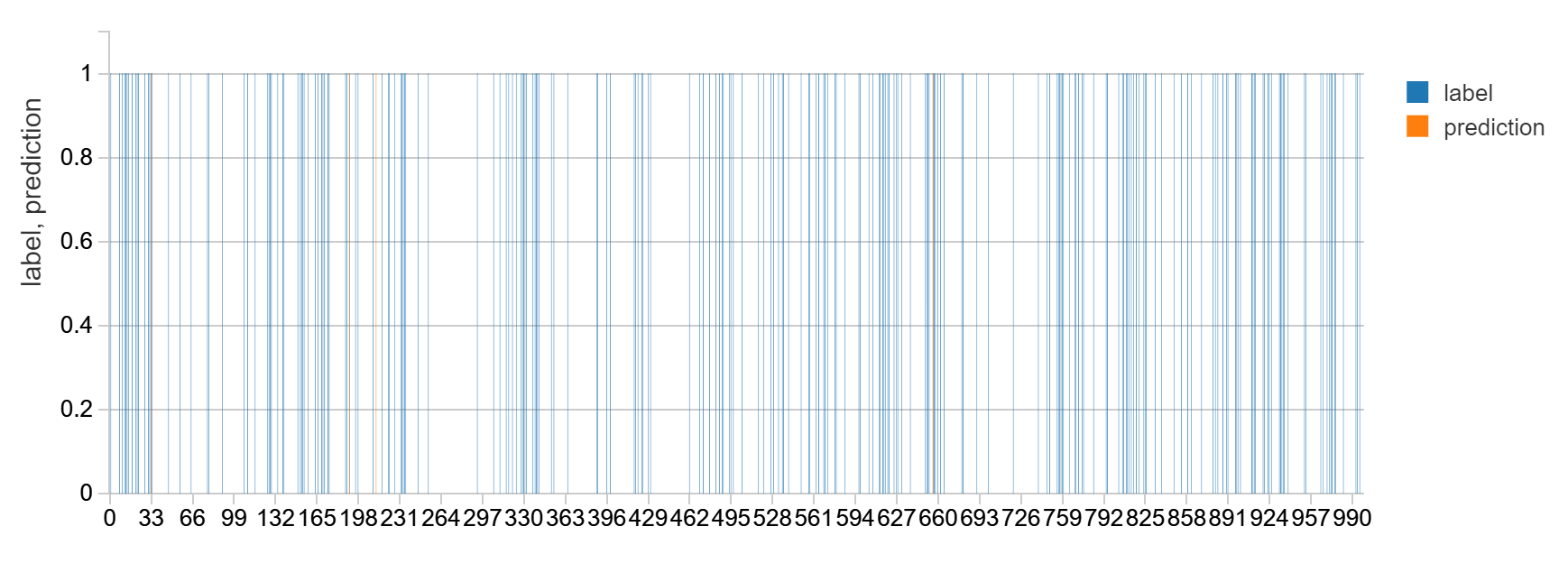
The min value is 0 in the above case because the GBT takes only (0,1) labels so we converted -1 values to 0.

**Label vs predicted label in GBT statistics:**



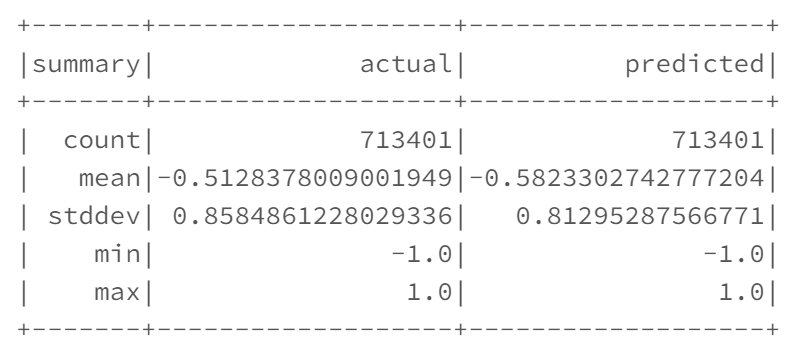
The mean value in the prediction is close to zero this means that the model could not predict properly the recommendations.

**Graph:**



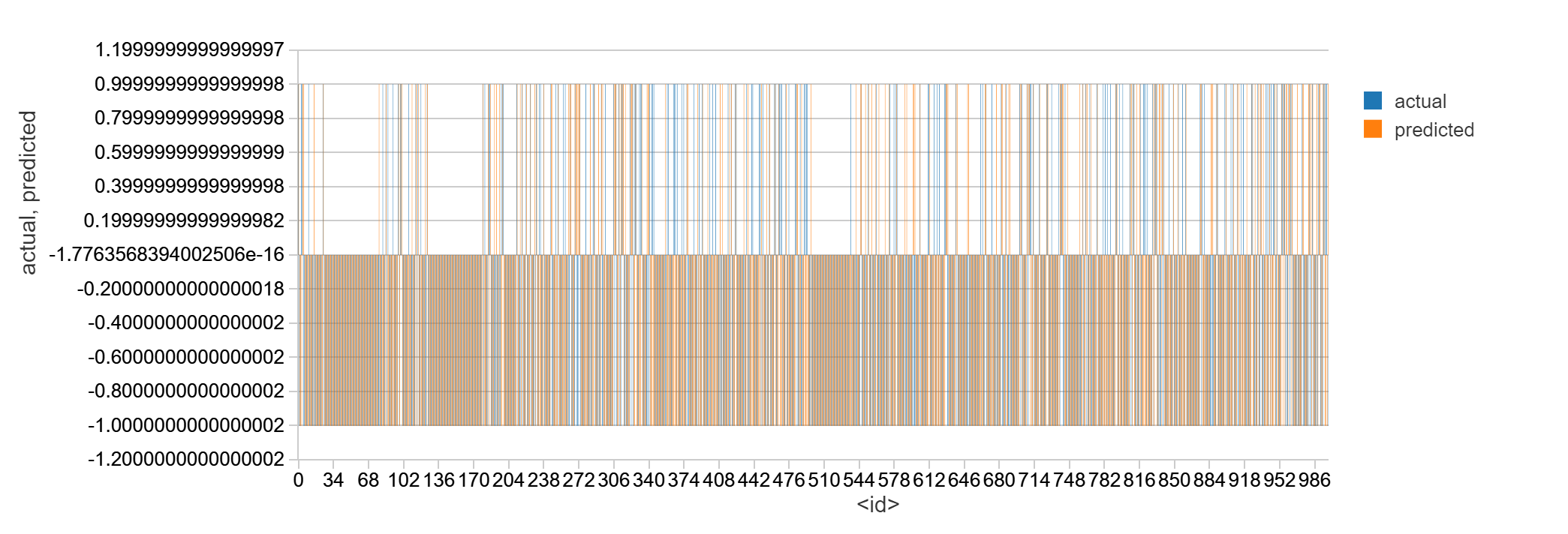
**Accuracy GBT: 0.721941**

**Statistics for actual vs predicted ALS:**

****

The mean in this case is almost close to the actual mean. Which means that the recommendation system has made fair amount of predictions for user interest in an item.

**Graph for ALS:**



Most of the data is overlapping in this case.

**Accuracy ALS: 0.770081623098**

**Conclusion:**

Our initial assumption that the keywords are only responsible for defining user interest in a product is false.

The GBT predicted the user interest wrongly by great amount when compared to ALS. Most of the cases GBT predicted that user is not interested in an item (close to 0 mean). There are other factors playing a significant role in the user’s interest in a product. ALS has the advantage over GBT by estimating all the features for a user’s interest in its matrices. ALS proves itself to be the best algorithm for the task.

**Contribution of Team members:**

One of us worked on the data preprocessing techniques and other on applying GBT method and ALS algorithm on the data.

**References:**

Main project: <https://www.kaggle.com/c/kddcup2012-track1#description>

ALS: <https://spark.apache.org/docs/latest/mllib-collaborative-filtering.html>

Gradient boosted tree regression: <https://spark.apache.org/docs/latest/ml-classification-regression.html#gradient-boosted-tree-classifier>

Features Extraction: <https://spark.apache.org/docs/latest/ml-features.html#word2vec>