Final Report of BAAI-ZHIHU Expert Finding Competition

Baodi Shan
lwshanbd@gmail.com
Shandong University
Qingdao, China

ABSTRACT

Knowledge sharing services have become one of the most important and popular applications on the global Internet. Therefore, how to connect knowledge, experts and users, and increase the willingness of experts to respond has become the central topic of knowledge sharing services. In this competition we will do something to solve this problem. In this process, I used assemble learning, XGboost and LGBM. At the same time, feature extraction is also a brightened dot of my work.

KEYWORDS

Knowledge Sharing, Datamining, XGboost, LGBM

1 INTRODUCTION AND BACKGROUND

This competition is one of in the Beijing Academy of Artificial Intelligence's 10 AI competitions in 2019.

Knowledge sharing services have become one of the most important and popular applications on the global Internet. In a knowledge sharing (or Q&A) community, the number of questions far exceeds the number of quality responses. Therefore, how to connect knowledge, experts and users, and increase the willingness of experts to respond has become the central topic of knowledge sharing services. This competition is designed to solve this problem.

Zhihu is a well-known knowledge sharing and Q&A community platform in China. Since its launch in 2011, Zhihu now has 220 million users, who ask hundreds of thousands of new questions or generate other content every day. Therefore, recommending right questions to most relative experts accurately and efficiently becomes an import task at Zhihu. The task includes mining and finding experts who are interested in the given questions and have enough expertise to answer them.

Figure 1: Logo of ZHIHU

2 HARDWARE AND SOFTWARE ENVIRONMENT

- Operating System: Ubuntu Server 16.04
- Graphics Processing Unit(GPU): NIVIDA GeForce 1080Ti
- Random-access Memory: 128GB
- Python Version: 3.7
- Integrated Development Environment(IDE): PyCharm, Jupyter Notebook

3 DATASET AND EVALUATION

3.1 Introduction of Dataset

The datasets released include the information of questions remained to be answerd by experts (or invitees), expert (or invitee) profiles, the experts (or invitees)' history of accepting invitation and answering behavior.

- Question Information. Incudes <question id, question publishing time, question's related topics, question's content text and question's description> on Zhihu's Q&A community.
- Expert (or invitee)'s answers, include <answer ID, question ID, author (or expert/invitee) ID, the text content of the answer, answer creation time, number of upvotes received, number of "bookmarks" by other users, number of "appreciates" by other users, number of comments> and etc.
- User profiling data, includes <user ID, gender, user activity frequency, followed topics, long-term interests, 'salt value'> and etc. 'Salt value', in general, is a credit indicator at Zhihu, which is decided by quantity and quality of the content the user has created, profile completeness, educational degrees, work experience and other qualifications, friendliness, contribution to the community and other factors.
- The information of <topics, token(words), single character embedding (64 dimensions)>.
- Invitation data within the previous month, include <question id, user id, invitation time, whether or not answer a question>.

3.2 Evaluation of Competition

The competition use AUC to evaluate the submitted file:

$$AUC = \frac{\sum_{iepositiveClass} rank_i - \frac{M(1+M)}{2}}{M \times N}$$

In the equation, M and N are positive and negative samples respectfully, rank is the location of the ith sample.

4 DATA PROCESSING

In this section, I will introduce how I process the data including reading data and feature extraction. As far as I am concerned, the most important part in this competition is the feature extraction rather than data training.

4.1 Data Loading

In my program, I used *pandas.read_csv* to load data. In the process of Uesr Infomation and Question Information ,my code is as follows

```
import pandas as pd
user_info = pd.read_csv('member_info_0926.txt', header=None,
    sep='\t')
user_info.columns = ['id','gender','keywords','number
    level','Hotness level','Registration Type','Registration
    platform','Frequency of visit','Binary feature A','
    Binary feature B',' Binary feature C',' Binary feature
    D',' Binary feature E','Classification feature A
    ','Classification feature B','Classification feature
    C','Classification feature D','Classification feature
    E','Salt value','Follow topics','Topic interested']
for col in user_info.columns:
    print(col, len(user_info[col].unique()))

question_info = pd.read_csv('question_info_0926.txt',
    header=None, sep='\t')
question_info.columns = ['id','time','title sig','title split
    ','des sig','des split ','topic ids']
for col in question_info.columns:
    print(col, len(question_info[col].unique()))
```

In the process of Invite Information ,my code is as follows:

```
invite_info = pd.read_csv(os.path.join(PATH,
    'invite_info_0926.txt'),
                      names=['question_id', 'author_id',
                          'invite_time', 'label'], sep='\t')
invite_info_evaluate = pd.read_csv(os.path.join(PATH,
    'invite_info_evaluate_1_0926.txt'),
                      names=['question_id', 'author_id',
                          'invite_time'], sep='\t')

invite_info['invite_day'] =
    invite_info['invite_time'].apply(lambda x:
    int(x.split('-')[0][1:])).astype(np.int16)
invite_info['invite_hour'] =
    invite_info['invite_time'].apply(lambda x:
    int(x.split('-')[1][1:])).astype(np.int8)

invite_info_evaluate['invite_day'] =
    invite_info_evaluate['invite_time'].apply(lambda x:
    int(x.split('-')[0][1:])).astype(np.int16)
invite_info_evaluate['invite_hour'] =
    invite_info_evaluate['invite_time'].apply(lambda x:
    int(x.split('-')[1][1:])).astype(np.int8)

invite_info = reduce_mem_usage(invite_info)
```

In the process of Answer Information ,my code is as follows:

```
answer_info = pd.read_csv(os.path.join(PATH,
    'answer_info_0926.txt'),
    names=['answer_id', 'question_id', 'author_id', 'answer_time',
        'content_sw_series', 'content_w_series','excellent',
        'recommend', 'round_table', 'figure', 'video','num_word',
        'num_like', 'num_unlike', 'num_comment','num_favor',
        'num_thank', 'num_report', 'num_nohelp', 'num_oppose'],
        sep='\t')
answer_info.head()

answer_info['content_sw_series'] =
    answer_info['content_sw_series'].apply(parse_list_2)
answer_info['content_w_series'] =
    answer_info['content_w_series'].apply(parse_list_1)
answer_info.head()

answer_info['answer_day'] =
    answer_info['answer_time'].apply(lambda x:
    int(x.split('-')[0][1:])).astype(np.int16)
answer_info['answer_hour'] =
    answer_info['answer_time'].apply(lambda x:
    int(x.split('-')[1][1:])).astype(np.int8)
del answer_info['answer_time']
gc.collect()

answer_info = reduce_mem_usage(answer_info)
```

4.2 Data Analysis

To extract the feature of the data better, we make some data analysis.

At first, we count the number of every attributes in the Answer Information and Question Information.

Table 1: Number of every attributes

Attributes	Number
uid	1931654
gender	3
keywords	1
number level	1
Hotness level	1
Frequency of visit	5
Binary feature A,B,C,D,E	2
Classification feature A	2561
Classification feature B	291
Classification feature C	428
...	...
Topic interested	1399721
qid	1829900
time	54617
title sig	1828611
title split	1786981
des sig	831554
des split	816606
topic ids	1166444

Then, we use make figures to seek the relationship between the attribute and "Answer or Not".Here are some figures.

There are three categories of gender characteristics, namely male, female and unknown. The below histogram shows that the distribution of males and females is very similar. Compared with the unknown distribution, there is a large difference. Obviously, this characteristic has better distinction.

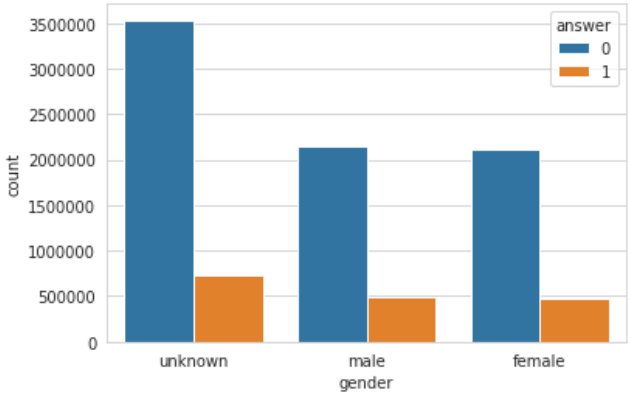

Figure 2: Gender

This feature has a total of 5 categories, [weekly, monthly, daily, new, unknown]. As can be seen from the histogram below, different categories have completely different distributions. This feature is obviously a very distinguishable feature.

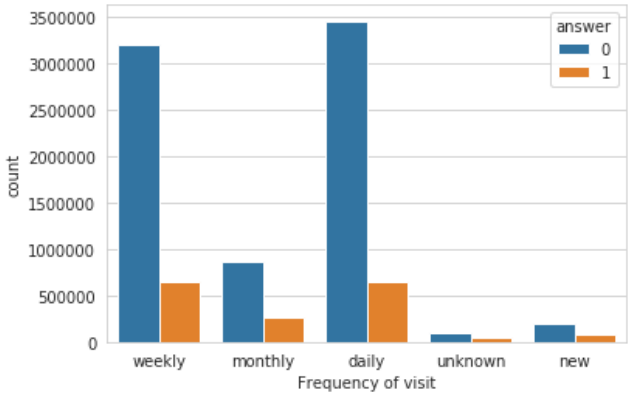

Figure 3: Frequency

This feature is a two-category feature, and the following figure shows that the feature has a good degree of discrimination (the remaining two-category and multi-category features are the same, and will not be repeated).

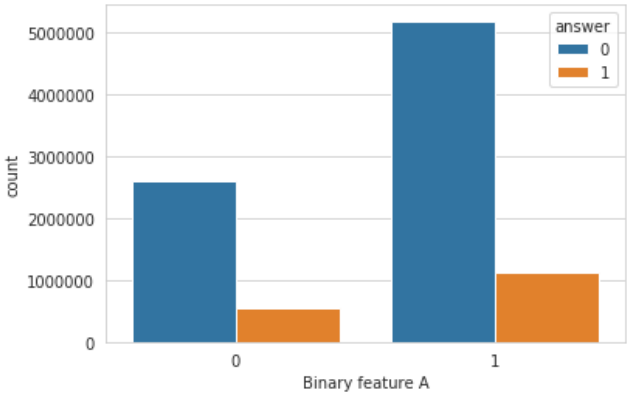

Figure 4: Binary feature A

The code of data visualization is as follows:

```
import matplotlib.pyplot as plt
import seaborn as sns

sns.set_style('whitegrid')

sns.countplot(x='gender',hue='answer',data=train)

#Out:<matplotlib.axes._subplots.AxesSubplot at
     0x2ab9b576f290>

sns.countplot(x='Frequency of visit',hue='answer',data=train)

#Out:<matplotlib.axes._subplots.AxesSubplot at
     0x2ab9a9974950>

sns.countplot(x='Binary feature A',hue='answer',data=train)

#Out:<matplotlib.axes._subplots.AxesSubplot at
     0x2ab9ac8e5250>
```

4.3 Feature Extraction

In this part we merge the member information and the question information and invite information firstly.The code is as follows:

```
train = pd.read_csv('invite_info_0926.txt', header=None,
     sep='\t')
train.columns = ['qid', 'uid', 'time','answer']
train = pd.merge(train, user_info, how='left', on='uid')
train = pd.merge(train, question_info, how='left', on='qid')
print(train.columns)
```

In this way, we get an initial train dataset, the index of it includes:

```
Index(['qid', 'uid', 'time_x', 'answer', 'gender', 'keywords',
     'number level',
      'Hotness level', 'Registration Type', 'Registration
           platform',
      'Frequency of visit', 'Binary feature A', 'Binary
           feature B',
```

```
    'Binary feature C', 'Binary feature D', 'Binary
        feature E',
    'Classification feature A ', 'Classification feature
        B',
    'Classification feature C', 'Classification feature D',
    'Classification feature E', 'Salt value', 'Follow
        topics',
    'Topic interested', 'time_y', 'title sig', 'title split
        ', 'des sig',
    'des split ', 'topic ids'],
   dtype='object')
```

Then, to utilize the feature of the member information of topics, we calulate the number, median number, max number and min number of member interested topics and some of other attributes. The code is as follows:

```
member_info['num_atten_topic'] =
    member_info['topic_attent'].apply(len)
member_info['num_interest_topic'] =
    member_info['topic_interest'].apply(len)


def most_interest_topic(d):
    if len(d) == 0:
        return -1
    return list(d.keys())[np.argmax(list(d.values()))]




member_info['most_interest_topic'] =
    member_info['topic_interest'].apply(most_interest_topic)
member_info['most_interest_topic'] =
    LabelEncoder().fit_transform(member_info['most_interest_topic'])

def get_interest_values(d):
    if len(d) == 0:
        return [0]
    return list(d.values())


member_info['interest_values'] =
    member_info['topic_interest'].apply(get_interest_values)
member_info['min_interest_values'] =
    member_info['interest_values'].apply(np.min)
member_info['max_interest_values'] =
    member_info['interest_values'].apply(np.max)
member_info['mean_interest_values'] =
    member_info['interest_values'].apply(np.mean)
member_info['std_interest_values'] =
    member_info['interest_values'].apply(np.std)




def process(df):
    return df.apply(lambda row: [row['topic_interest'][t]
        for t in row['topic_interest_intersection']],axis=1)

pool = mp.Pool()
chunk_list = split_df(invite_id_qm, 100)
ret = pool.map(process, chunk_list)
invite_id_qm['topic_interest_intersection_values'] =
    pd.concat(ret)
gc_mp(pool, ret, chunk_list)




invite_id_qm['num_topic_attent_intersection'] =
    invite_id_qm['topic_attent_intersection'].apply(len)
invite_id_qm['num_topic_interest_intersection'] =
    invite_id_qm['topic_interest_intersection'].apply(len)



invite_id_qm['topic_interest_intersection_values'] =
    invite_id_qm['topic_interest_intersection_values'].apply(lambda
    x: [0] if len(x) == 0 else x)
invite_id_qm['min_topic_interest_intersection_values'] =
    invite_id_qm['topic_interest_intersection_values'].apply(np.min)
invite_id_qm['max_topic_interest_intersection_values'] =
    invite_id_qm['topic_interest_intersection_values'].apply(np.max)
invite_id_qm['mean_topic_interest_intersection_values'] =
    invite_id_qm['topic_interest_intersection_values'].apply(np.mean)
invite_id_qm['std_topic_interest_intersection_values'] =
    invite_id_qm['topic_interest_intersection_values'].apply(np.std)

feats = ['author_question_id',
    'num_topic_attent_intersection',
    'num_topic_interest_intersection',
    'min_topic_interest_intersection_values',
    'max_topic_interest_intersection_values',
    'mean_topic_interest_intersection_values',
    'std_topic_interest_intersection_values']
feats += []
member_question_feat = invite_id_qm[feats]
```

At last, we encode other features.Related function code is as follows:

```
def parse_str(d):
    return np.array(list(map(float, d.split())))

def parse_list_1(d):
    if d == '-1':
        return [0]
    return list(map(lambda x: int(x[1:]), str(d).split(',')))

def parse_list_2(d):
    if d == '-1':
        return [0]
    return list(map(lambda x: int(x[2:]), str(d).split(',')))

def parse_map(d):
    if d == '-1':
        return {}
    return dict([int(z.split(':')[0][1:]),
        float(z.split(':')[1])] for z in d.split(','))
```

5 CLASSIFIER

In this competition, I used the XGboost Classifier and LGBM Classifier. In this section, I will introduce these two Classifiers and introduce how I used them.

5.1 XGboost

XGBoostis an open-source software library which provides a gradient boosting framework for C++, Java, Python. It works on Linux, Windows,and macOS. From the project description, it aims to provide a "Scalable, Portable and Distributed Gradient Boosting (GBM, GBRT, GBDT) Library". It runs on a single machine, as well as the distributed processing frameworks Apache Hadoop, Apache Spark, and Apache Flink. It has gained much popularity and attention recently as the algorithm of choice for many winning teams of machine learning competitions.

Compared to other traditional classifier, we have to transform the data from array form to Dmatrix form. The code of it is as follows:

```
import xgboost as xgb
from sklearn.metrics import accuracy_score
import scipy

dtrain = xgb.DMatrix( X_train, label=y_train)
dval=xgb.DMatrix( X_val, label=y_val)
```

There are many parameters should be adjusted, here are some important paramaters:

- *objective*
  Specify the learning task and the corresponding learning objective. The objective options are below:
  reg:squarederror
  reg:squaredlogerror
  reg:logistic
  ...
  binary:logistic
  multi:softmax In this competition, I will definitely choose the"binary:logistic",because I need to give the probability of binary classfication qusetion.
- *gamma*
  Gamma is the minimum loss reduction required to make a further partition on a leaf node of the tree. The larger gamma is, the more conservative the algorithm will be. In my program,I choose 0.1as my gamma
- *max depth*
  Maximum depth of a tree. Increasing this value will make the model more complex and more likely to overfit. 0 is only accepted in lossguided growing policy when tree-method is set as hist and it indicates no limit on depth. Beware that XGBoost aggressively consumes memory when training a deep tree.During the process of my preparing for the competition,I found this is the most influential paramater in all paramaters. Finally, I choose 8 as my max depth.
- *lambda*
  L2 regularization term on weights. Increasing this value will make model more conservative.I choose 2 as my max lambda.
- *tree method*
  XGBoost supports approx, hist and gpu hist for distributed training. Experimental support for external memory is available for approx and gpu hist.Because my server has a NVIDIA GPU, I choose the "gpu hist",whose training is much faster than CPU.

Code of this part is as follows:

```
num_round = 5000
params = {
    'booster': 'gbtree',
    'objective': 'binary:logistic',
    'gamma': 0.1,
    'max_depth': 8,
    'lambda': 2,
    'subsample': 0.8,
    'colsample_bytree': 0.8,
    'min_child_weight': 3,
    'silent': 0,
    'eta': 0.002,
    'seed': 1000,
    'tree_method':'gpu_hist',
    'predictor':'gpu_predictor',
    'eval_metric': ['logloss', 'error', 'auc'],
    'verbose_eval':True,


}



bst = xgb.train( params, dtrain, num_round,
     evals=[(dval,'evl')])
dtest = xgb.DMatrix(test[feature_cols])
ans1 = bst.predict(dtest)
```

5.2 LGBM

LightGBM is a gradient boosting framework that uses tree based learning algorithms. It is designed to be distributed and efficient with the following advantages:

- Faster training speed and higher efficiency.
- Lower memory usage.
- Better accuracy.
- Support of parallel and GPU learning.
- Capable of handling large-scale data.

Code of this part is as follows:

```
model_lgb = LGBMClassifier(n_estimators=2000, n_jobs=-1,
     objective='binary', seed=1000, silent=True)
model_lgb.fit(X_train, y_train,
            eval_metric=['logloss', 'auc'],
            eval_set=[(X_val, y_val)],
            early_stopping_rounds=50)
ans2= model_lgb.predict_proba(test[feature_cols])[:, 1]
```

6 RESULT AND SUMMARY

In the end, my online accuracy rate reached **81.542%**, ranking **62th** among all the contestants, and ranking **1st** among the students

who chose this competition in the "Information Retrieval and Data Mining" class.

By taking part in this competition, I can skillfully use LGBM and XGboost classifiers. And it improved my inderstanding of machine learing and data mining. Although finally I haven't won any awards, I think I still benefit a lot from *BAAI-ZHIHU Expert Finding Competition*

7 ACKNOWLEDGMENTS

I would like to extend my sincere gratitude to my supervisor, **Prof. Jianhua Yin**, for your instructive advice and useful suggestions on my thesis. I am deeply grateful of your help in the completion of this thesis.