Project Report

Cross-Device Entity Linking Challenge

CS249 Winter 2017

Team Name: BTF
Team Member:
Lizhi Zeng (304593058)
Dui Lin (504759948)
Pei Jiang (604685278)
Zhiming Zhuang (204593426)
Yang Guo (104588741)

1. Introduction

For online advertising, companies help their clients market products and services to the right audiences of online users. However, users now have multiple choices to access the internet with smart devices, and even the same device could be shared by many users. In order to show advertisements to target user, user identity plays an essential role in the success of an online advertising company/platform. Unless a service supports persistent user identities (e.g. Facebook Login), the same user on different devices is viewed independently. Therefore, the ability to accurately identify the user at the device level and link the same users across multiple devices is desirable.

The Challenge is a part of the CIKM 2016 and continues the CIKM Cups series co-arranged as part of the ACM CIKM conference. Here, we are given a dataset provided by DCA (Data-Centric Alliance), based on which we try to generate a list of candidate user pairs that are predicted to be the same person. The dataset contains an anonymized browse log for a set of anonymized user IDs representing the same user across multiple devices. And obfuscated site URLs and HTML titles are also given.

In this report, we describe our solution for the challenge. We formulated the task as a binary classification problem. After generating all possible pairs which contains both potential matched pairs and random pairs, we try to label each pair with 0 or 1 representing random pair or matched pair respectively. The three most essential parts are the feature engineering, reduced candidate selection and model training. The overview framework in shown in Figure 1-1.

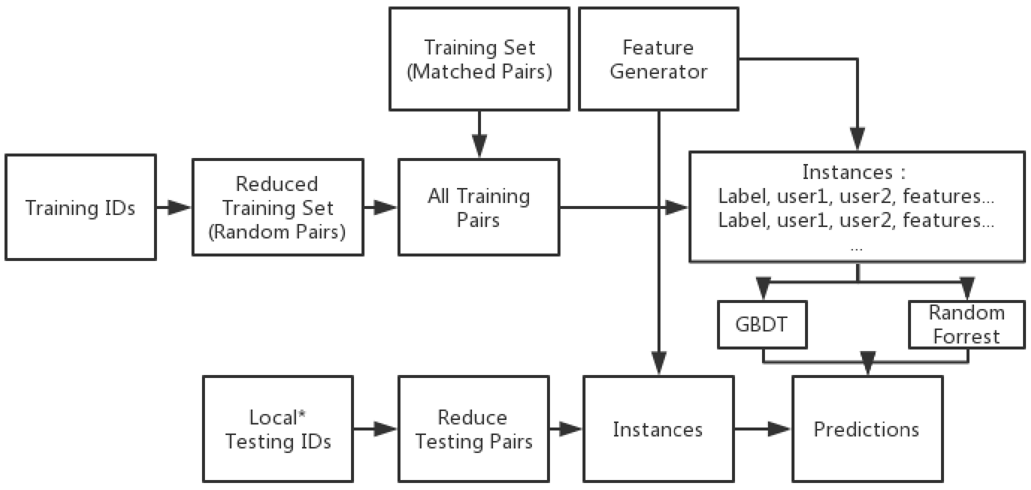

Figure 1-1. Framework of our approach

The remainder of our report is organized as follows. In Section 2, we review the dataset from different perspectives. Then we describe our feature engineering approach in Section 3. In Section 4, we discuss the reduced candidate selection method. In Section 5, we introduced the models we use and describe some property of our models. And we present the prediction and comparison of different models in Section 6, followed by the conclusion and outlook in Section 7. Also, we attached the reference,data link and important intermediate result at the end.

2. Data Overview

2.1 Dataset Description

We are offered 4 datasets in the cross-device entity linking challenge.

Name	**Item Number**	**Description**
facts.json	339,405	A browsing log containing a list of events for a specific userID: *fid* is an eventID; *ts* is a timestamp; and *uid* is a userID. e.g. *{facts: [{fid, ts}, {fid, ts}, ...], "uid"}*
URLs.csv	27,398,114	A mapping from an *fid* to the hashed *URL*. e.g. (fid, aaa/bbb/ccc)
titles.csv	8,485,859	A mapping from an *fid* to the hashed *HTML title*. e.g. (fid, AAA BBB CCC)
train.csv	506,136	A set of matching userIDs for a supervised cross-device matching model training. e.g. (uid1, uid2)

Table 2-1. Name of Dataset with its item number and description

During the process, we firstly map each user to its related URLs and HTML titles which are connected by sharing the same fid in facts.json.

2.2 Missing Values

During the following feature engineering process, the missing value problems occurs when we tried to match each user in the training set with its corresponding HTML titles according to their shared fid. Some event number does not have a matched HTML title in the titles.csv. Therefore, we drop these pairs when preparing features to ensure the training process.

2.3 Dataset Statistics

Other statistical information that might be helpful to give an overview on the datasets is listed as below.

Statistics	**Value**
Number of users	339,405
Number of fids	12,148,535
Maximum number of fid/user	2000
Minimum number of fid/user	2
Average number of fid/user	197
Median number of fid/user	106
Number of unique domain names	282,613
Number of URL(1st level)	230,297
Number of matched pairs for training	506,136
Number of matched pairs to predict	215,307
Number of users in the train set	240,732
Number of users in the test set	98,255

Table 2-2. Dataset statistics

3. Feature Engineering

Feature engineering is an essential step of data preprocessing. It is designed to extract useful information from huge size of raw data. By looking through the extracted information, observed patterns could be contribute to the model implementation. And the choices of certain features greatly affect the model performance in the later part.

Since feature engineering is almost the most important part for data mining projects, we designed comprehensive features based on the browsing logs. Each instance is a pair of users with label 1 if the user pair is a match and label 0 if not. We divide our features into four groups which are presented separately in the following sections.

3.1 General Similarity

For each matched pair of users, we assume that they are similar in web surfing behavior. In other words, for a pair of matched users, they will have more common web browsing history than unmatched pairs. Therefore, refer to Lian's and Xie's papers, we apply a general similarity metric to evaluate the similarity of two users' browsing behavior.

To calculate the the general similarity, we first collect URL title, Event ID and HTML title for each user. We consider the URL title, event ID and HTML title as terms and users as documents. After cleaning and reconstructing data from given files, we have three bags of words. Based on a bag of words, we calculate TF-IDF weight for each term and build up a TF-IDF matrix on user domain:

$$w_i^u = \frac{n_i}{\sum_k n_k} \times \log \frac{|\{\mathrm{doc}_j\}|}{|\{\mathrm{doc}_j : \mathrm{n}_i \in \mathrm{doc}_j\}|} \tag{1}$$

and measure the similarity between two users with cosine similarity measurement:

$$Sim(\mathrm{u},\mathrm{v}) = \frac{w^u \cdot w^v}{\|\mathrm{w}^u\| \cdot \|\mathrm{w}^v\|} \tag{2}$$

where (u, v) stands for a use pair.

To be noticed, since not all users in dataset have the complete information (e.g. Not all users have the HTML title records). Therefore, we eliminate users who do not have complete information. After cleaning and reconstructing data, the final statistical result of unique users in training set is 240,036, and the final number of matched pair in training set is 503,312.

- **DocSim**

The document similarity (DocSim) regards the HTML titles for a user as a word. After calculating the TF-IDF weights, we received 182,471 different terms. The dimension of HTML titles TF-IDF matrix is 240,036*182,471. This statistical result indicates that users have shared many common HTML titles, and this feature should be important for training our classifier.

- **FidSim**

The event ID similarity (FidSim) regards the Fid for a user as a word. After calculating the TF-IDF weights, we received 1,887,864 different terms. The dimension of HTML titles TF-IDF matrix is 240,036*1,887,864. This statistical result indicates that users do not share much common event ID, and this feature will be distributed sparsely in the space.

- **URLSim**

The URL similarity (URLSim) regards the URL titles for a user as a word. Since the number of URL for each user is large and our time is limited, we only choose the first level of the URL title. After calculating the TF-IDF weights, we received 121,651 different terms. The dimension of HTML titles TF-IDF matrix is 240,036*121,651. This statistical result indicates that users have shared many common URL titles, and URLSim should be important for training our classifier.

3.2 Common Count

- **FidComCnt and URLComCnt**

We count the number of common fid and URLs between the two users.

3.3 Time Related Features

- **HourCor**

As is often the case, users may have some temporal patterns in their online behaviors. For example, some users are active at early morning, some are at noon, while others are active at midnight. Therefore, we calculate the Pearson correlation coefficient based on the time distribution of two users:

$$HourCor(\mathrm{u},\mathrm{v}) = \frac{\sum_{i=1}^{24}(\mathrm{t}_i^u - \overline{t^u})(\mathrm{t}_i^v - \overline{t^v})}{\sqrt{\sum_{i=1}^{24}(\mathrm{t}_i^u - \overline{t^u})^2}\sqrt{\sum_{i=1}^{24}(\mathrm{t}_i^v - \overline{t^v})^2}} \tag{3}$$

where 24 means there are 24 hours in a day and t means how many times a user click URLs in an hour.

- **DayCor**

Similar to HourCor, but here we calculate the Pearson correlation coefficient based on day distribution (from Monday to Sunday), so there will be 7 days and t means how many times a user click URLs in a day.

- **MonthCor**

Similar to HourCor, but here we calculate the Pearson correlation coefficient based on month distribution (from January to December), so there will be 12 months and t means how many times a user click URLs in a month.

- **HourCE**

Consider the same concern with HourCor, but here we use cross entropy as the metrics:

$$HourCE(\mathrm{u},\mathrm{v}) = -\sum_{i=1}^{24} t_i^u \log(\mathrm{t}_i^v) \tag{4}$$

- **DayCE**

Similar to HourCE, but here we calculate the cross entropy based on day distribution (from Monday to Sunday).

- **MonthCE**

Similar to HourCE, but here we calculate the cross entropy based on month distribution (from January to December).

- **HourTemporalDist**

We use original tempral distributions as features. For features in hour granularity, we use a 24-dimension vector to record the hoURLy activity amount and use euclidean metric to calculate the temporal distance between two users:

$$HourTemporalDist(\mathrm{u},\mathrm{v}) = \sqrt{\sum_{i=1}^{24}(\mathrm{t}_i^u - \mathrm{t}_i^v)^2} \quad (5)$$

- **DayTemporalDist**

Similar to HourTemporalDist, we use a 7-dimension vector (from Monday to Sunday) to record the daily activity amount and use use euclidean metric to calculate the temporal distance between two users.

- **MonthTemporalDist**

Similar to HourTemporalDist, we use a 12-dimension vector (from January to December) to record the daily activity amount and use use euclidean metric to calculate the temporal distance between two users.

- **FirstDateGap**

We use the interval between the first dates of the two users when they have their first recorded online events. We believe that the same user will have similar behaviors from the perspective of login time. The specific date can be found under 'ts' tag in the format of unix time. We use utcfromdatetime function to parse all the unix times to microseconds, calculate and generate the feature in the unit of seconds.

- **LastDateGap**

Similar to FirstDateGap, we use the interval between the last dates of the two users when they have their last recorded online events. We believe that the same user will have similar behaviors from the perspective of logout time. Here, we also calculate and generate the feature in the unit of seconds, which could satisfy the accuracy and efficiency.

- **OverlapDay**

To verify the variance of active dates between two online footprints which are estimated to come from the same person. We count the number of dates both the two users are active. Considering that all the timestamps are retrieved from the year of 2016, we ignore the year part in the timestamp and only compare the month and date.

- **Skewness**

To make sure the data will not fall into data skew, we also need to introduce a parameter to reflect skewness in a pair of users. Here, we define skewness as the ratio of shorter lifespan to longer lifespan:

$$Skewness(\mathrm{u},\mathrm{v}) = \frac{Min(\mathrm{lifespan(u)},\mathrm{lifespan(v)})}{Max(\mathrm{lifespan(u)},\mathrm{lifespan(v)})} \quad (6)$$

3.4 Weighted URL Related Features

The basic idea here is trying to distinguish URLs consisting of different degrees of importance. For example, google.com is commonly visited among different users. However, visiting the same personal blog among PC and mobile application strongly indicates a user matching.

Here, we want to design features which can describe what kind of URLs that two user share. Ideally, there are some URLs that appear more often in matched pairs than random pairs. These URLs are considered as more critical while differentiating matched pairs from random pairs.

In order to find those URLs, for each URL we calculate the ratio of the probabilities that it appears simultaneously in a matching pair to that in a random user pair.

$$urlRank(\mathrm{h}) = \frac{|\,number\ of\ matched\ pairs\ containing\ h\,| \,/\, |\,number\ of\ matched\ pairs\,|}{|\,number\ of\ random\ pairs\ containing\ h\,| \,/\, |\,number\ of\ random\ pairs\,|} \quad (7)$$

Note that here we only take the first level of URL of each event. Table 3-1 lists the top 10 first level URLs with their urlRank numbers.

Rank	**First Level URL**	**urlRank Value**
1	e7b5bc4559be5f7	21593.740728447163
2	9e897b044f199a37	19277.934727029886
3	2223dcc4007201d7	18035.681176600756
4	cb50fa4925dfd5e4	17805.63422281758
5	fd523864c0e45fe7	17621.59665979104
6	2b0a8e1424a90724	15413.145903472583
7	070951333610 64d	13940.845399260275
8	814d6485f3c5b051	12698.591848831144
9	669c0dd1f855d5d7	11824.413424455086
10	c4c2edb80d9eca55	11318.310126132106

Table 3-1. Top 10 key first level URLs with their urlRank

Based on the Key URL ranking, we generate two fine-grained features considering the occurrence of key URLs.

- **KeyURLDist**

We measure the distribution of common URLs of each pair. To be specific, we divide the top 4000 key URLs into 40 buckets, with each bucket containing 100 URLs. For each user-user pair, we count the number of their common URLs in each bucket respectively.

- **TopURLHit**

Since the top key URLs shows an extremely high probability of matching users, in this feature, we use a 500-dimensional indicator vector to record whether the top 500 key URLs exist in the common space of the two users.

3.5 Feature Engineering Summary

Since feature engineering is usually the most important factor for a data mining model, we have designed comprehensive features from different levels in order to get a satisfying result. To sum up, we generate 558 features in total, which can be mapped into 4 categories. The details are shown in table 3-2.

Feature Category	**Feature Name**	**Feature Dimension**
General Simulariy	DocSim	3
	FidSim	
	URLSim	
Common Count	FidComCnt	2
	URLComCnt	
Time Related	Hour/Day/MonthCor	13
	Hour/Day/MonthCE	
	Hour/Day/MonthTemporalDist	
	FirstDayGap	
	LastDayGap	
	OverLapDay	
	Skewness	
Weighted URL Related	TopURLHit	540
	KeyURLDist	

Table 3-2. Feature Engineering Overview

4. Reduced Candidate Selection

4.1 Training Set Candidate Selection

As we have discussed in section 3.1, some users do not have completed information. After we eliminate those users, we have 240,073 unique users. Since we only have 506,136 matched pair, if we generate all possible negative pairs (240,073*240,072/2) to train, it will be unbalanced for positive pairs and negative pairs. Therefore, we apply sub-sampling to generate negative pairs. For a user u, we randomly choose another user v, and pair them together as (u,v) if this pair does not exist in the train list. We have also considered the situation that (u,v) and (v,u) are the same pair. For each user, we repeat the process 100 times and we receive 23,287,009.

However, it is still too large because positive_set:negative_set = 1:46. Therefore, we continue the sub_sampling work by selecting only the first 10 pairs among every 100 pairs. Finally, we receive total 2,819,161 training pairs (including both positive and negative). And positive_set:negative_set = 1:4.6, which is more reasonable.

4.2 Testing Set Candidate Selection

According to the baseline document, number of users to predict is 98409. Testing all possible pairs, which is of size 98409*(98409-1)/2, sets too heavy workload on a single machine. Considering the huge amount of data, we firstly performed candidate selection based on TF-IDF vector to select likely candidates. To construct TF-IDF vectors, we construct tokens for URLs in the facts dataset.

For each user, we generate k nearest neighbors and add them to the list of prospective candidates. Since the candidate set determined by k is closely related to the accuracy of the prediction, we did experiments with different k values. In case k is large, we filter the candidates by the distance to the cluster center to restrict the dataset within the acceptable size for testing. To be specific, we tried k=100 with distance less than 1, k=50 with distance less than 1, k=40 with distance less than 1.2 and k=30. Considering computational efficiency and accuracy, we choose k=30. And we generate a list of 2446941 possible pairs for testing.

5. Models

For now, our training set and test set are ready. To train our classifier, we apply three advanced classification models which are Gradient Boosting Classifier, Random Forest Classifier and Support Vector Machines.

5.1 Model Overview

In this section, we provide a brief discussion on the algorithm behind the two models.

The first classifier we have chosen is random forest. Random forest is also an ensemble learning method for classification. It operates by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes of the individual trees. Random forest is correct for decision trees' habit of overfitting to their training set.

Our second classifier is gradient boosting. Gradient boosting is a machine learning technique for regression and classification problems, which produces a prediction model in the form of an ensemble of weak prediction models, typically decision trees. It builds the model in a stage-wise fashion like other boosting methods do, and it generalizes them by allowing optimization of an arbitrary differentiable loss function. In our gradient boosting classifier, we choose logistic regression as the loss function.

5.2 Model - Random Forest

Firstly, we use the 558-dimensional input as stated in section 3 and 4 to train our random forest model. It takes about an hour to finish training the model excluding the time to load the input data after preprocessing and feature engineering.

For the original random forest model, we explore the feature importance attribute in order to continue reducing comparatively useless features, aiming at accelerating the process of training and predicting in the next stage.

Noticing only a few features gain an obvious advantage over other features, we pick the first 58 features and discard the remaining 500 columns to compose a new input training set. Again, we train a revised random forest model with a much shorter time, and we look through the features’ importance to obtain an overview of how these remaining features work. Figure 5-1 shows the top 10 important features of both random forest models with a normalized score, while table 5-1 shows the specific features among the top 10. In practice, we train two models with the first 58 features, one with 10 estimators and the other with 100 estimators. The details are revealed in the following section.

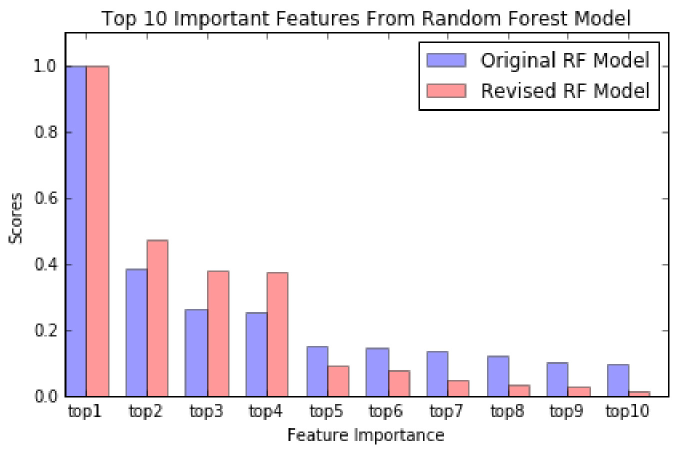

Figure 5-1. Top 10 important feature from two random forest models

Top	Feature Name - Original RF	Feature Name - Revised RF
1	URLSim	FidComCnt
2	URLComCnt	URLSim
3	Top(1100-1200)URLHit	URLComCnt
4	DayCE	HtmlSim
5	Top(3400-3500)URLHit	FidSim
6	Top(3500-3600)URLHit	HourTemporalDist
7	Top(500-600)URLHit	DayTemporalDist
8	Top(1900-2000)URLHit	MonthTemporalDist
9	Top(3000-3100)URLHit	HourCor
10	Top(1700-1800)URLHit	MonthCor

Table 5-1. Top 10 feature names from two random forrest models

5.3 Model - Gradient Boosting

A benefit of using gradient boosting is that after the boosted trees are constructed, it is relatively straightforward to retrieve importance scores for each attribute. The more an attribute is used to make key decisions with decision trees, the higher its relative importance. The importance is calculated explicitly for each attribute in the dataset, allowing attributes to be ranked and compared to each other. In scikit learn, we can invoke *model.feature_importances_* to see the importance scores of each feature. The summary of gradient boosting classifier is shown in Table 5-3-1.

Input Dimension	Train Start Time	Train End Time
(2819161, 558)	03/15 02:27:21	03/16 10:42:07
(2819161, 58)	03/16 16:43:56	03/16 16:45:48

Table 5-3-1. Gradient Boosting Classifier Summary

We first use 558 features to train our model. And do feature selection based on our trained gradient boosting classifier. The Top 11 attributes are shown in Table 5-3-2.

Top	Feature Name-Original GRD	Feature Name-Revised GRD
1	FidComCnt	FidComCnt
2	URLSim	URLSim
3	FidSim	FidSim
4	MonthTemporalDist	MonthTemporalDist
5	HourTemporalDist	HourTemporalDist
6	FirstDate	FirstDate
7	LastDate	LastDate
8	HtmlSim	HtmlSim
9	HourCor	HourCor
10	DayCor	DayCor

Table 5-3-2. Top 10 feature names from two random forest models

We can see from the table that the top 10 features of original gradient boosting classifier (i.e. trained by 558 features) is similar to the top 10 features of revised one. This indicates that these 10 features are significant in identifying a user's entity. This result is reasonable and can reflect users' web browsing behavior. It is easy to image that same user may visit the same websites often and surf internet at the similar time in a day.

6. Model Evaluation

We evaluated using F1 score by comparing the ground-truth userID matching with the predicted results. In statistical analysis of binary classification, the F1 score (also F-score or F-measure) is a measure of a test's accuracy. It considers both the precision p and the recall r of the test to compute the score: p is the number of correct positive results divided by the number of all positive results, and r is the number of correct positive results divided by the number of positive results that should have been returned.

$$F_1 = 2 \cdot \frac{1}{\frac{1}{recall} + \frac{1}{precision}} = 2 \cdot \frac{precision \cdot recall}{precision + recall} \tag{8}$$

Specifically, for Precision we will count the number of pairs predicted correctly among the pairs and for Recall we will count the number of predicted correctly pairs out of all ground-truth matching userIDs available in the validation set posted on the competition website.

6.1 Evaluation Benchmark

According to CIKM Cup official website (https://competitions.codalab.org/competitions/11171#results), we download the top 20 performances statistics among all participants, which could be a benchmark for our own predicted results. Note that the rankings of precision and recall may not match, for example, the team with No. 1 precision score has No. 7 recall score. Table 6-1 shows the sorted precision scores and recall scores respectively.

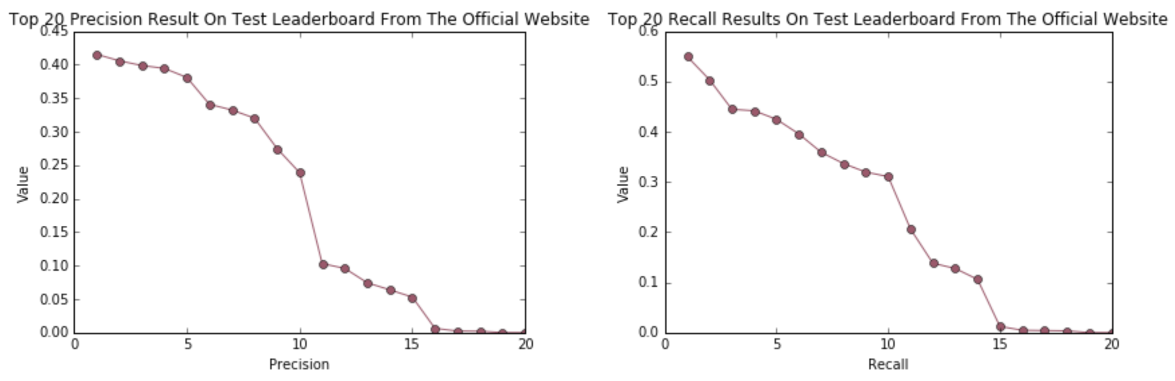

Figure 6-1. Top 20 Precision and Recall Performances Among All Participants

6.2 Evaluation - Random Forest

The Random Forest model can predict the possibility for each pair being labeled as class 1. By setting the threshold of labeling the candidate to different values, we can get the following graphs of precision, recall and F1 score with respect to threshold.

We first trained a random forest with 558 features and 10 estimators. However, as it's showed in Figure 6-2, the highest F1 score of the classifier is around 0.16 and the F1 score keeps dropping as the threshold increases, which indicates the poor performance of the classifier. So, we did experiments with more estimators to improve the accuracy. Besides, according to the importance of features , we found out that the last 500 features are of little importance. Therefore, we get rid of the last 500 features to improve the efficiency. The resulting precision, recall and F1 score of model with 58 features with 10 and 100 estimators are as follows. The highest F1 scores are 0.1759 and 0.1822 respectively, both at threshold 0.67.

Figure 6-2 indicates the three criteria for evaluating models, where rf_58_10 means random forest trained with 58 features and 10 estimators, rf_58_100 means random forest trained with 58 features and 100 estimators, rf_558 means random forest trained with 558 features in Figure 6-2's legend. Meanwhile, Table 6-1 shows the statistical comparisons between three random forest models using the same notation.

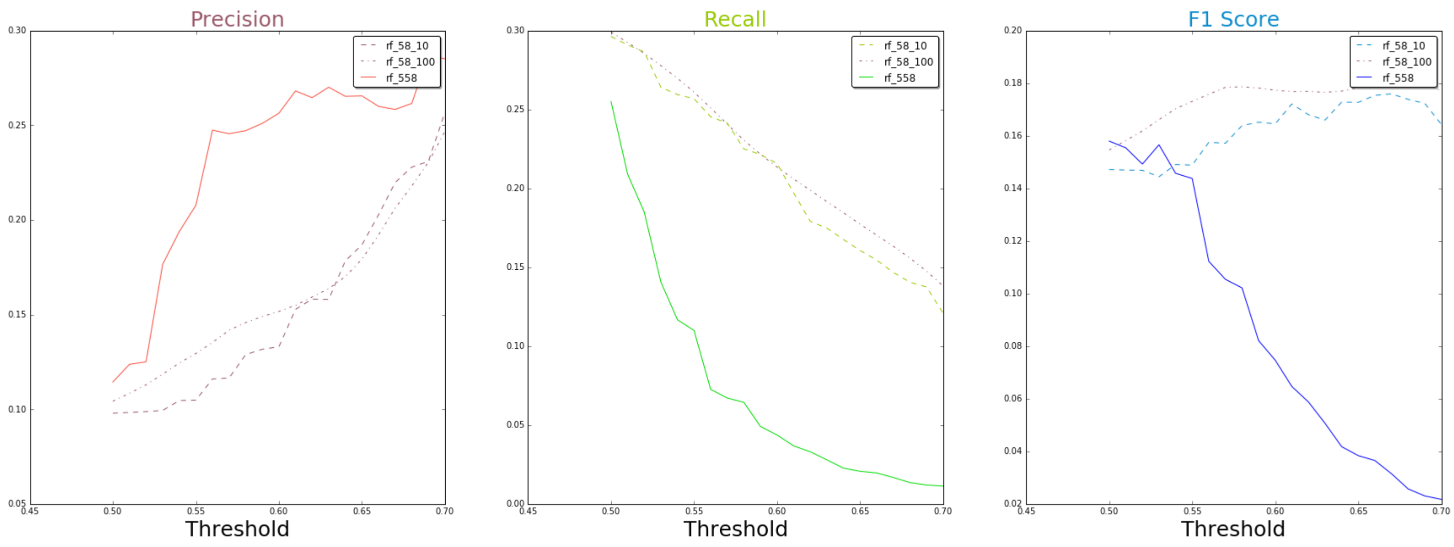

Figure 6-2. precision, recall and F1 scores from random forest models

Classifier	Best F1 Score	Threshold	Precision	Recall
RF_558	0.157983521991	0.5	0.11442076143	0.255110145049
RF_58_10	0.175944029456	0.67	0.219745510961	0.146702150882
RF_58_100	0.182276683202	0.67	0.206204005043	0.163324926733

Table 6-1. Summary of Random Forest Classifier

6.3 Evaluation - Gradient Boosting

Gradient boosting classifier returns the probability that a user i belongs to class j. Usually, the classifier will classify the use to the class with a probability greater than 0.5 in default. However, we want a more precise prediction. Therefore, we set up threshold to do a second selection on predicted result. Since the highest probability of predicting 1 is around 0.71, the upper bound of threshold are set to 0.7. Summary of the two gradient boosting classifiers are shown in Table 6-2. And the evaluation of our results (precision, recall, F1 score) are shown in Figure 6-3 and 6-4.

Classifier	Best F1 Score	Threshold	Precision	Recall
Original	0.2326	0.67	0.1810	0.3255
Revised	0.2327	0.67	0.1811	0.3255

Table 6-2. Summary of Gradient Boosting Classifier

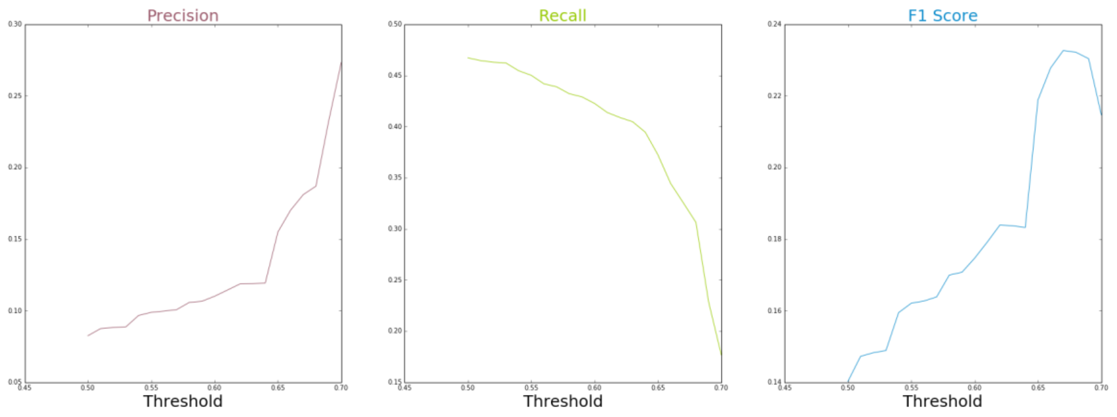

Figure 6-3. precision, recall and F1 scores of gradient boosting model using 558 features

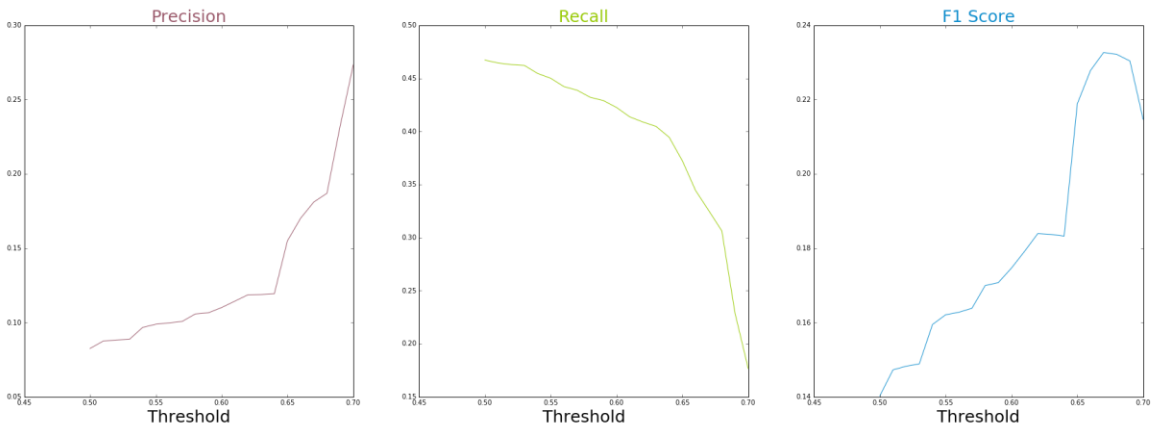

Figure 6-4. precision, recall and F1 scores of gradient boosting model using 58 features

7. Conclusion and Outlooks

In general, our task is to find the same user across multiple devices. There are three primary components that we focus on: feature engineering, candidate selection, and model training with selection. We can see from the previous sections 6.2 and 6.3 that gradient boosting classifier outperforms random forest in identifying the test user pairs. This is a great experience for us to handle a real-world problem and deal with real world data. The most challenge part in this project is that we have to solve the problem of unbalanced dataset. We have done a lot of research and choose sub sampling method to process the training set and KNN to pre-process the testing set based on time and computation resource.

Our results outperform half of the competitors in the leaderboard of this competition, though we are limited in time and computation resources. Here we discuss the future outlooks based on the issues we encountered during the project. First of all, the sampling approaches can be improved by modifying the parameter priority as weighted random sampling, which is more fault-tolerant. Also, we encourage the followers to try more sampling approaches like oversampling, etc. Secondly, the number of estimators for classification can be modified, which will help to train and build a more precise model. Furthermore, regarding the feature engineering from URL files, we only consider the cases of the first level URLs for the sake of reducing computation burden. We expect that the followers can integrate deeper level URLs to provide more precise information of matching pairs and increase the precision of models. Finally, the weight of each feature can also be redistributed and we could redesign the metric of generating features to improve our results.

8. Acknowledgements

This project was supported by the Department of Computer Science at University of California, Los Angeles. We thank Professor Wei Wang and Teaching Assistants Ariyam Das and Xinxin Huang from CS249 who provided excellent course contents and expertise in data mining algorithms that greatly inspired our work.

We would also like to show our gratitude to Jianxun Lian from University of Science and Technology of China for sharing his pearls of wisdom especially his ideas in data sampling with us.

References

[1] J. Lian, X. Xie, Cross-Device User Matching Based on Massive Browse Logs: The Runner-Up Solution for the 2016 CIKM Cup, 1st ed. arXiv, 2016.

[2] Minh C. Phan, Yi Tay, Tuan-Anh Nguyen Pham, Cross Device Matching for Online Advertising with Neural Feature Ensembles : First Place Solution at CIKM Cup 2016, arXiv, 2016.

[3] R. D´ıaz-Morales. Cross-device tracking: Matching devices and cookies. arXiv preprint arXiv:1510.01175, 2015.

[4] T. Chen and C. Guestrin. Xgboost: A scalable tree boosting system. In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016, pages 785–794, 2016.

[5] J. Walthers. Learning to rank for cross-device identification. In 2015 IEEE International Conference on Data Mining Workshop (ICDMW), pages 1710–1712. IEEE, 2015.

APPENDIX

I. Distribution of works

Zhiming and Dui did the project presentation and helped with the survey paper. Lizhi, Yang and Pei worked on proposal and project report.

In the project, we divided the features into 4 categories. Lizhi is responsible for features in ‘general similarity’ and ‘common count’. Zhiming prepared features related to time, Hour/Day/Month Cor/CE/TemporalDist. And Dui got the remaining features in ‘time related’ ready. Yang and Pei generated key URL rankings and ‘weighted URL’ related features.
In the training process, Lizhi focused on Gradient boosting, while Zhiming and Yang did random forest respectively. And we all got involved in summarizing our work and polishing the report and survey paper.

II. Data Link

The Challenge is a part of the CIKM 2016 and continues the CIKM Cups series co-arranged as part of the ACM CIKM conference. Our data files are downloaded from the following address:
https://competitions.codalab.org/competitions/11171#learn_the_details-data2

III. Attachment

1.Feature names and corresponding indices:

0: DayCE

1: DayCor

2: DayTemporalDist

3: HourCE

4: HourCor

5: HourTemporalDist

6: MonthCE

7: MonthCor

8: MonthTemporalDist

9: FirstDate

10: LastDate

11: OverLapDay

12: Skewness

13: HtmlSim

14: FidSim

15: URLSim

16: FidComCnt

17: URLComCnt

18: # top100 urls

19: # top100-200 urls

20: # top200-300 urls

……

57: # top 3900-4000 urls

58: 1st url or not

59: 2nd url or not

…...

557: 500th url or not

2. Feature importance of random forest:

58 features with 100 estimators:

```
[  1.23851600e-02   2.94731216e-03   7.02698823e-03   2.11738241e-02
   1.77873525e-03   1.03910108e-02   3.23760618e-03   9.43998730e-04
   2.15744959e-03   3.39130396e-07   3.63855938e-07   4.45036642e-04
   3.51726908e-06   1.49753183e-01   1.14009820e-01   1.67894939e-01
   2.10492376e-01   1.41872020e-01   9.79187112e-03   5.91434613e-03
   8.37450539e-03   1.70405276e-03   3.63772881e-03   3.35571249e-07
   3.90601988e-03   0.00000000e+00   4.83264063e-08   3.32402123e-03
   4.28844268e-08   9.29173035e-03   2.75431289e-07   0.00000000e+00
   0.00000000e+00   5.89630133e-04   0.00000000e+00   5.44244170e-04
   7.29715317e-04   3.65765426e-03   0.00000000e+00   4.91889288e-03
   5.44887499e-04   1.71288969e-03   0.00000000e+00   0.00000000e+00
   0.00000000e+00   7.39923392e-04   1.22350016e-07   0.00000000e+00
   6.98527249e-03   1.23167001e-02   0.00000000e+00   0.00000000e+00
   3.58559194e-02   1.99910983e-02   0.00000000e+00   0.00000000e+00
   1.19455184e-03   1.77598404e-02]
```

3. Feature importance of gradient boosting:

558 features:

0., 0.0124784 , 0. , 0. , 0.01300488, 0.02876588, 0., 0.00361638, 0.02945426, 0.0236834 ,0.02220781, 0. , 0, 0.01404667, 0.09787751, 0.1115358 , 0.64332901, 0., 0., 0…….(zeros following)

58 features:

0. 0.01247915 0. 0. 0.01300452 0.02876517 0. 0.00361621 0.02945268 0.02368258 0.02220714 0. 0. 0.01404619 0.09789805 0.11153365 0.64331466 0…..(zeros following)