ABSTRACT
As the number and variety of smart devices increase, users may use myriad devices in their daily lives and the online activities become highly fragmented. Building an accurate user identity becomes a difficult and important problem for advertising companies. The task for the CIKM Cup 2016 Track 1 was to find the same user across multiple devices. This paper discusses our solution to the challenge. It is mainly comprised of three parts: comprehensive feature engineering, negative sampling, and model selection. For each part we describe our special steps and demonstrate how the performance is boosted. We took the second prize of the competition with an F1-score of 0.41669.

Keywords
User Linking; User Profiling; Cross-Device Behavior Analysis; CIKM Cup

1. INTRODUCTION
With the rapid development of smart devices, users now have myriad choices to connect to the Internet for daily activities. A user may do shopping with his/her smart phone, primary work on a laptop, and watch movies on a tablet. Unless a service supports persistent user identities (e.g. Facebook Login), the same user on different devices is viewed independently. It results in companies having to deal with weak user identities at device level. To perform sophisticated user profiling especially for online advertising, it is important to link the same users across multiple devices and integrate his/her digital traces together.

At the Conference on Information and Knowledge Management (CIKM) 2016, the Data-Centric Alliance (DCA) provided a dataset for cross-device entity linking challenge. The dataset contained an anonymized browse log for a set of userIDs representing the same user across multiple devices. For each browse log, DCA provided the obfuscated site URL and HTML title. Some of the linked users were released as the training set. The participants need to identify the remaining matching user across multiple devices. Submissions were evaluated using F1 measure (a harmonic mean of precision and recall).

In this paper, we describe our solution which placed 2nd at the competition. We formulated the task as a binary classification problem. Generally, it is simple, intuitive, and extremely effective. The three most essential parts are the feature engineering, negative sampling, and model selection. The framework is shown in Figure 1. Feature engineering is usually the most important factor for a data mining model. To achieve a satisfying score, we have designed comprehensive features from different levels. For the majority of data mining competitions, gradient boosted machine is the best single model and the ensemble of various models can further improve performance. We also consider this tip but conduct ensemble in a different way: we use the gradient boosted decision tree as the core classification model and use the logistic regression model to filter candidates. Since the complete candidate set is \( N \times N \) which is too large in space, we have to do negative sampling. We find that the choice of negative instances significantly influences the performance of the model.

The remainder of this paper is organized as follows. In Section 2 we briefly review the data set. Then we describe our feature engineering approach in Section 3. In Section 4 and 5 we discuss our negative sampling algorithm and model selection, respectively. The online judging is presented in Section 6, followed by the conclusion in Section 7.

2. DATASET OVERVIEW
There are four data files in total provided for the compiti-
tion. The first one is facts.json which contains users’ browsing logs. Each browsing log contains a list of events for a specific user, including fid (which can be regarded as an event ID), timestamp, and user id. All the IDs are anonymized. Another two files contain the mapping from an fid to the URL and HTML title, respectively. The last file offers a set of matching user IDs for training. The basic statistics of the dataset are shown in Table 1. We denote URL level as the length of the path, e.g., “bing.com” is level-1, “bing.com/images” is level-2.

3. FEATURE ENGINEERING

We formulated the user-matching task as a binary classification problem. Since feature engineering is usually 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. The feature set can be divided into three pillars as follows:

3.1 General Similarity

We assume that if two activity traces on two devices belong to the same user, then the traces will share some common websites or be similar in content. Thus, we design general similarity metrics from the perspective of words, events, URLs, and time:

**DocSim**: For each user we collect a bag of words from the title of HTML the user has visited. Based on this bag of words we calculate the words’ weight in terms of TF-IDF and regard it as the user’s document profile. For two users, their document similarity (DocSim) is measured as the cosine similarity between the document profiles:

\[ w_i^u = \frac{n_i}{\sum_k n_k} \times \log \frac{|\{doc_j : n_i \in doc_j\}|}{|\{doc_j : n_i \in doc_j\}|} \]  
\[ w_u =\langle w_i^u, w_j^u, \ldots, w_m^u \rangle \]  
\[ DocSim(u, v) = \frac{w_u \cdot w_v}{\|w_u\|\|w_v\|} \]  

**FidSim**: Similar to DocSim, however here we regard each event ID as a word and calculate the similarity based on the event document profiles.

**URLSim**: Similar to DocSim, however here we regard each URL as a word and calculate the similarity based on the URL document profiles. Since we consider 4 kinds of URL levels as shown in Table 1, we get 4 values from URLSim.

**FidComCnt, URLComCnt**: We count the number of common fid and URLs between the two users.

**HourCor**: We assume that users may have some temporal patterns in their online behaviors. For example, some users are active at midnight, while some users get up early in the morning. Thus we calculate the Pearson correlation coefficient based on the time distribution of two users:

\[ HourCor(u, v) = \frac{\sum_{i=1}^{24} (t_i^u - \bar{t})(t_i^v - \bar{t})}{\sqrt{\sum_{i=1}^{24} (t_i^u - \bar{t})^2} \sqrt{\sum_{i=1}^{24} (t_i^v - \bar{t})^2}} \]  

**HourCE**: Consider the same concern with HourCor, but here we calculate the cross entropy as the metrics:

\[ HourCE(u, v) = -\sum_{i=1}^{24} t_i^u \log t_i^v \]  

**DayCor**: Similar to HourCor, but here we calculate the Pearson correlation coefficient based on day distribution (from Monday to Sunday).

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

**MonthCor, MonthCE**: Similar to HourCor and HourCE but from the point of month.

**OverlapDay**: We count the number of dates both the two users are active.

**Skewness**: The ratio of shorter lifespan to longer lifespan:

\[ Skewness(u, v) = \frac{\min(lifespan(u), lifespan(v))}{\max(lifespan(u), lifespan(v))} \]  

![Figure 1: The framework of our solution.](image-url)
3.2 Key URLs

The above feature pillar is called General Similarity because they are coarse-grained. For example, we calculate the number of mutual URLs between user u and user v, but we don’t know which particular URLs they are. Visiting bing.com is commonly happening among different users, while a common visit to a personal homepage strongly indicates a user matching. To this end, we plan to design more fine-grained features in this pillar which can describe what kind of URLs the two users share. We find that there are some URLs that appear more often in positive pairs than negative pairs. We assume that those URLs are key URLs that can differentiate matching users from dis-matching users. In order to find out those URLs, for each URL that can differentiate matching users from dis-matching users.

\[
\text{RatioLift} = \frac{\text{#matching pairs containing } h}{\text{#random pairs containing } h} \quad (7)
\]

Table 2 lists the top 10 key URLs. We can observe that these URLs are much more likely to appear in positive pairs than in negative pairs. There are about 4000 URLs with lift ratio above 2000. We categorize key URLs into 7 groups by the lift ratio: top 100, top 1000, top 2000, top 3000, top 4000, top 5000, and the others. For each user pair we count the number of key URLs in each group as features.

3.3 Footprints

We plan to design further fine-grained features in this pillar. We want to include the detailed activities of the users and meanwhile avoid overfitting.

KeyURLDist: We sort the key URLs by their lift ratio and define 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 URLs shows an extremely high probability of matching users, in this feature, we use a 500-dimension indicator vector to record whether the top 500 key URLs exist in the common space of the two users.

TemporalDist: In the General Similarity pillar, we calculate the Pearson correlation similarity and cross entropy between two users’ temporal (hour/day/month) distributions. Here we use the original temporal distributions as features. For example, for features in hour granularity, we use a 24-dimension vector to record the hourly activity amount.

3.4 Feature Evaluation

| URL (level-1) | RatioLift |
|--------------|-----------|
| 426db1edc2529037/9d6b15ac43b3eb9c | 5.686,956 |
| 449c90845cf01f8/b32ca6f60250833b | 3.913,043 |
| 449c90845cf01f8/77c813065f72822 | 3.900,000 |
| c0f203845417df4/16e752084757385eb | 2.739,130 |
| 449c90845cf01f8/3c3df5b4cf02638a2 | 2.647,826 |
| 449c90845cf01f8/10548343d5806a2 | 2.647,826 |
| 09bb529d5b61c1b/e0e89a73c6372082 | 2.478,260 |
| 51667bbf4f1509987/0804738d6681160c | 2.269,565 |
| 09bb529d5b61c1b/ac4f7a4715b4762 | 2.230,434 |
| 09bb529d5b61c1b/77cc413057b22ef2 | 1.996,652 |

Table 3: Performance evaluation with feature incrementation. Row General-Sim means using feature pillar 1 only. +Key URLs means using feature General-Sim and Key URLs. Finally +Footprints means using all the three feature pillars.

| Features   | AUC   | Recall | Precision | F1    |
|------------|-------|--------|-----------|-------|
| General-Sim | 0.8756 | 0.4299 | 0.4968    | 0.4445|
| +Key URLs  | 0.8890 | 0.4991 | 0.5034    | 0.4513|
| +Footprints| 0.9383 | 0.5613 | 0.6906    | 0.6193|

We reserve 1000 users from the training file as our local validation set\(^2\). There are 3076 matching pairs in the local validation set. Table 3 shows how the performance is improved when we add more fine-grained features. Adding footprints features significantly improves all the evaluation metrics, which demonstrate that fine-grained features play an essential role in user profiling. Table 4 lists the top 10 most important features according to the build-in feature evaluation functionality of gradient boosting machine\(^3\). It further demonstrates that the footprint features carry the most discriminative information.

4. NEGATIVE SAMPLING

There are a total of 339,405 unique users in the browse log, where the number of users in the training set is 240,732. As described in the last section, for each instance (user-user pair), we extract 621 features. We randomly output 10,000 instances and found that the file size is about 17.74MB. In this way, enumerating the whole user-user pairs will result in 57,951,895,824 instances in the training file, which requires about 100TB in space. Thus we have to do negative instances sampling. We denote U as the user set, M as the matching pairs, S to be the sampled training instances. We

Algorithm 1 Iterative Negative Sampling

Input: U, M, n and k
Output: S

\[\text{model} \leftarrow \text{NULL}\]

for \(i = 0 \rightarrow k - 1\) do

\[\text{S} \leftarrow \emptyset\]

for \(u \in U\) do

if \(\text{model} = \text{NULL}\) then

Randomly sample \(n\) users from U and add the \(n\) pairs to S

else

Select top \(\frac{n}{2}\) users from U according to the \(\text{model}\) and add to S

Randomly sample \(\frac{n}{2}\) users from U and add to S

end if

end for

\[\text{S} \leftarrow \text{S} \cup M\]

re-train \(\text{model}\) based on \(\text{S}\)

end for

return S

\[\text{Output: S}\]

\[\text{Input: U, M, n and k}\]

\[\text{Algorithm 1 Iterative Negative Sampling}\]

\[\text{A common practice is that we need to generate k validation sets in order to perform significance test on experimental results. Due to time limit we just skip this step.}\]
Table 4: Top 10 most important features.

| Feature Name      | Split Gain |
|-------------------|------------|
| KeyURLDist02      | 1.0        |
| HourCorrelation   | 0.4499     |
| FidSim            | 0.4246     |
| KeyURLDist01      | 0.4221     |
| URLSim Level-1    | 0.3478     |
| TopURLHit10       | 0.3183     |
| KeyURLDist00      | 0.3137     |
| OverlapDay        | 0.2910     |
| KeyURLDist07      | 0.2494     |
| KeyURLDist08      | 0.2389     |

Table 5: Performance evaluation for iterative negative sampling algorithm.

| Sampling method | AUC   | Recall | Precision | F1   |
|-----------------|-------|--------|-----------|------|
| Random          | 0.8433| 0.4195 | 0.4301    | 0.4247|
| INS             | 0.9383| 0.5613 | 0.6906    | 0.6193|

5. MODEL SELECTION

We compare the performance of several models, including Gradient Boosted Decision Tree (GBDT), Logistic Regression (LR), Shallow Neural Network (SNN), and SVM (with linear kernel). Figure 2 shows the results, from which we can observe that GBDT significantly outperforms the other three models. It is in accordance with expectation because gradient boosting machine is the state-of-the-art classification model as reported from most of data mining competitions.

A golden rule for achieving the top rank in data mining competitions is that we need to train various models and make ensembles. However, it is usually time consuming to find an optimal way to ensemble. We didn’t join the competition until the last week of the end of the competition. So we propose a simple but turns out to be efficient way to ensemble: from the training data we train two model, LR and GBDT. We use the LR model to select 100 candidates for each user in the test set. After this we can get a smaller test set. Then we use the GBDT model to make predictions on the smaller test set. Table 6 shows that this simple approach greatly improves the accuracy of the prediction. One possible explanation is that there are many negative instances which non-linear (tree) model such as GBDT could not differentiate from positive instances. However, linear models like logistic regression happens to work well on this part. Being aware of this, the results shown in Figure 2 are unfair to LR because for the test set we already apply LR to select top 100 candidates in order to reduce the test space. Due to time limits we did not spend more effort on the model ensemble. However, the aforementioned result implies that there are still opportunities to make improvements.

6. ONLINE EVALUATION

Every time we make improvements to local evaluations, the corresponding online F1-score also improves. It indicates that our framework does not cause overfitting and the local test set is extracted appropriately. There are a total of 215,307 true pairs in the test set. One small trick is that

Table 7: Top 5 teams on final leader board.

| Rank | F1     | Precision | Recall |
|------|--------|-----------|--------|
| 1    | 0.42038| 0.39875   | 0.4449 |
| 2    | 0.41669| 0.39444   | 0.4416 |
| 3    | 0.41370| 0.40042   | 0.4279 |
| 4    | 0.40168| 0.36591   | 0.4452 |
| 5    | 0.36110| 0.33227   | 0.3954 |

Table 6: Performance of GBDT without and with LR filtering.

| Model               | AUC   | Recall | Precision | F1   |
|---------------------|-------|--------|-----------|------|
| No LR filtering     | 0.7864| 0.3551 | 0.4046    | 0.3782|
| With LR filtering   | 0.9383| 0.5613 | 0.6906    | 0.6193|

https://www.kaggle.com/wiki/PastSolutions
Algorithm 2 Select Instances for Submission

Input: predicted test pairs $T$, and parameters $n, k, r$.  
Output: submission set $P$  

\[ P \leftarrow \emptyset \]

for each user $u$ do

for $i = 0 \rightarrow k - 1$ do

retrieve the $i$-th top predicted user $v$ for $u$

if $\text{Ranking}(T, (u, v)) < n \times r$ then

Add $\langle u, v \rangle$ to $P$ if not exists

end if

end for

end for

return $P$

since F1-score is a trade-off between recall and precision, we don’t need to submit too many predicted instances in order to achieve a peak F1 score. We only included about 100,000 instances in our final submission file. The final post processing algorithm is shown in Algorithm 2. Besides top $n$ pairs, for each user in the test set we also select the top $k$ candidates whose global rank is not far away from $n$. We ended up with 2nd place on the leader board. Table 7 lists the top 5 teams’ scores. The top 3 teams’ final scores are very close.

7. CONCLUSION

In this paper, we describe our solution for the CIKM Cup 2016 User Linking Challenge at which we took the second place of the competition. It has three primary components that we focus on: the feature engineering, negative sampling, and model selection. Since time was limited when our approach was conceived, there are still many possible approaches which we have not tried yet. For example, learning-to-rank [2] is a promising approach to this competition; and we can also apply other ensemble methods such as stacking.

8. RELATED WORK

The topic of CIKM Cup 2016 (track 1) is very similar to the ICDM Cup 2015 Drawbridge Cross-Device Connections, except that the two events provide different types of data for mining. Among the winning solutions [10] [6] [3], learning-to-rank and binary classification are the two most popular paradigms. [10] points out that learning-to-rank is more suitable for cross-device linking problem because for each entity, we don’t need the absolute value of its probability in matching with another entity, instead what we need is the relative ranking according to the target entity. [3] combines several techniques, such as semi-supervised learning and bagging, to further boost the performance. Since it is not feasible to generate a full entity-to-entity pair set, down-sampling is used by all the winning solutions. However, they down-sample the candidates by some particular rules. In this paper, we propose a negative sampling method which selects candidates iteratively with a weak learner.

Cross-device user matching is also related to link prediction [8] [11] [11]. From the graph-theoretical perspective, users can be regarded as nodes and a user-matching can be modeled as an edge between the two corresponding nodes. Thus the task is to predict the missing links in the graph. [5] surveys several well studied link mining tasks and methods. In this paper, we model the task as a binary classification problem for simplicity. In the next steps we will study how to make breakthroughs using graph models.

9. REFERENCES

[1] M. Al Hasan, V. Chaoji, S. Salem, and M. Zaki. Link prediction using supervised learning. In SDM06: workshop on link analysis, counter-terrorism and security, 2006.

[2] C. J. Burges. From ranknet to lambdarank to lambdamart: An overview. Technical report.

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

[4] J. H. Friedman. Greedy function approximation: a gradient boosting machine. Annals of statistics, pages 1189–1232, 2001.

[5] L. Getoor and C. P. Diehl. Link mining: a survey. ACM SIGKDD Explorations Newsletter, 7(2):3–12, 2005.

[6] G. Kejela and C. Rong. Cross-device consumer identification. In Proceedings of the 2015 IEEE International Conference on Data Mining Workshop (ICDMW), ICDMW ’15, pages 1687–1689, Washington, DC, USA, 2015. IEEE Computer Society.

[7] J. Lian, X. Xie, and G. Sun. Winning the Second Place of IJCAI-15 Repeat Buyers Prediction Contest: a Feature Engineering Approach. https://github.com/Leavingseason/Competitions/blob/master/Jianxin/IJCAI_Cup2015.pdf, 2015.

[8] D. Liben-Nowell and J. Kleinberg. The link-prediction problem for social networks. Journal of the American society for information science and technology, 58(7):1019–1031, 2007.

[9] G. Liu, T. T. Nguyen, G. Zhao, W. Zha, J. Yang, J. Cao, M. Wu, P. Zhao, and W. Chen. Repeat buyer prediction for c-commerce. In Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD ’16, pages 155–164, New York, NY, USA, 2016. ACM.

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

[11] Z. Yin, M. Gupta, T. Weninger, and J. Han. A unified framework for link recommendation using random walks. In Advances in Social Networks Analysis and Mining (ASONAM), 2010 International Conference on, pages 152–159. IEEE, 2010.