Cross-Device Entity Linking Challenge

Binary classification to pairwise model

Nam Khanh Tran
ntran@l3s.de

ABSTRACT

In this paper, we summarize the 5th place solution for the CIKM cup 2016 and show that learning-to-rank models provide an improvement over the binary classification for the cross-device entity linking task. We also point out several possible directions to improve the results.

1. INTRODUCTION

The CIKM Cup 2016 Track 1: Cross-Device Entity Linking Challenge was hosted by CodeLab from August 5th to October 5th. The competition attracted 155 registered participants and 46 actual competitors.

The goal of this competition is to identify the same users across multiple devices (predict new edges in the anonymized userID graph) using the browse log and associated meta-data. The participants have to submit the most likely pairs of anonymized userIDs (edges in the matching graph), where both userIDs are associated with the same user. One userID might match to many userIDs because a user might have more than 2 devices.

The evaluation metric used for the challenge was the F_1 score. One important noted here is that participants can submit any number of pairs of anonymized userIDs, thus optimizing the number of prediction pairs might be needed.

In this paper, we describe our approaches for tackling the linking problem. In the first approach, we consider it as a binary classification problem: *given two userIDs, classify them into to the class 1 or the class 0, indicating they associate with the same user or not, perspectively.* In the second approach, we treat the linking problem as learning-to-rank task:*give an userID, rank other userIDs according to their possibilities referring to the same user*

2. DATASET

The organizers provided a collection of browse logs and associated meta-data in four different files: *facts.json*, *urils.csv*, *titles.csv* and *train.csv* [1]. Some dataset statistics are shown in Table 1

Total users	339,405
Number of testing users	98,255
Training matching pairs	506,136
Testing matching pairs	215,307
Unique tokens in titles	8,485,859
Unique tokens in urls	27,398,114

Table 1: Dataset statistics

As mentioned in the task description, the training and testing sets of users are separated and about 0.5% "noise" users are added to the testing set resulting a total of 98,673 users.

From the provided dataset, we have some interesting observations which provide us some clues for our next steps:

- About 8000 (1.5%) training pairs of userIDs do not share any events/actions
- This number for sharing domains between userIDs in the training pairs are about 2000 (0.4%)

3. BUILDING A TRAINING AND PREDICTING SET

3.1 Negative examples

In the training set, about 500,000 positive pairs of userIDs are provided but no negative pairs, thus we need to generate negative examples for our learning algorithm.

In principle, for each userID of 240,732 userIDs in the training set, we can generate a large number of negative pairs (approximately 40B). But it is not feasible for any learning algorithm. Thus, we need a method for sampling a good set of negative pairs from training userIDs.

We observed that an user usually creates same events across different devices such as watching videos on Youtube, accessing to social platforms Facebook/Twitter, etc. For each userID u, we created a list of related userIDs $L(u)$ which share one or more events/facts with u. Then, for each positive pair of userIDs (u_1, u_2), we created six negative pairs, three for the userID u_1 and three for the userID

[1]The detailed description of each file can be seen on the competition website https://competitions.codalab.org/competitions/11171

CIKM Cup 2016 USA

DOI: 10.1145/1235

u_2 which are randomly selected from $L(u_1)$ and $L(u_2)$, respectively. For the userID u whose $L(u)$ is empty, they are randomly chosen from training userIDs.

To this end, we obtained a total of 3,5 millions training examples.

3.2 Prediction set

Since the number of userIDs in test set are large (approximately 98 thousands), it is not feasible to use all possible pairs of the userIDs as candidate pairs. Hence, we propose a blocking method for generating a smaller set of possible pairs of userIDs.

Based on the observation that users tend to make the same actions across devices, we generate a set of candidate pairs of usersIDs where two userIDs share one or more events/facts. To this end, we obtained 56 millions candidate pairs of userIDs from the test set.

In addition, we attempted to use another blocking method to generate the prediction set of pairs, in which a pair of userIDs is considered as a candidate if they both refer to one or more domains. However, this method ends up with a huge number of pairs; thus, because of the computation cost, we decided to use the event-based blocking method.

4. APPROACHES

In this section, we describe two approaches for tackling the cross-device linking problem. The first one considers the problem of cross-device entity linking as binary classification and makes use of the state of the art classification algorithms, whereas the second one considered it as learning to rank problem and used a pairwise approach and a greedy algorithm for generating the final prediction.

4.1 Learning features

Given a pair of userIDs, we extracted a set of features based on the browse logs and associated meta-data: *facts, domain, title, time.*

4.1.1 Fact features

This type of feature is generated based on the fact/event information itself. Each user is considered as a document and the facts that she/he takes are considered as words of the document. Then, a number of features are extracted including jaccard, tf-idf similarities.

In addition, the topic distributions of each document (user) are also learned by topic modeling methods such as Latent Dirichlet Allocation (LDA)[2], the similarity measures including Hellinger, cosine, jaccard are used to compute based on the topic distributions. The doc2vec[3] similarity is also calculated as one of the semantic features.

4.1.2 Domain and Title features

The domain and title features are extracted in the similar way as the fact features, each user is considered as a document but the words of the document are represented by the domains and titles instead of the facts.

4.1.3 Time features

Similarly to the previous features, we represented each user as a sequence of days which he/she has any activity, and then jaccard, rf-idf features are extracted.

[2]http://mallet.cs.umass.edu/topics.php

[3]https://radimrehurek.com/gensim/models/doc2vec.html

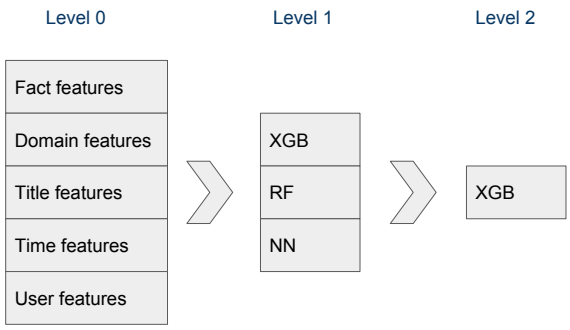

Figure 1: Ensemble approach for cross-device entity classification

In addition, we assume that users tend to use different devices in different time. For example, they use desktop/laptop in the working time e.g. from 9am to 12am, and mobile phone during the lunch time and their tablets in the evening. Thus, we use the distance between the time (hour) of userIDs as another time feature.

4.1.4 Hybrid features

This type of feature considers a combination of time and other meta-data into account such as time features for each fact or each domain.

In total, we generated 75 different features, and use them for our learning algorithms.

4.2 Prediction as classification

Figure 1 describes an overview of our ensemble approach for tackling the cross-device entity problem. We start with 75 features extracted from each pair of userIDs in the layer 0. These features are used as the inputs for the classification algorithms in layer 1, whose outputs are used as the inputs for the algorithm in layer 2.

The training dataset in layer 0 is split into learning and validation sets in which 2/3 examples are used for learning. The parameters of algorithms in layer 1 are tuned using the K folds strategy on the learning set and the best configurations are used to generate predictions of examples on the validation set.

The prediction scores from the algorithms in the layer 1 are considered as meta-features for the algorithm in layer 2. The best parameters are tuned on the validation set using the K folds method, then used to generate the final predictions for the pairs in the prediction set.

In the layer 1, we explore several supervised learning algorithms including Neural Network (NN), Extreme Gradient Bossting (xgboost) and Random Forest, while xgboost is used in the layer 2.

4.3 Prediction as ranking

In this section, we describe another approach to tackle the problem, in which we consider the cross-device linking problem as the ranking problem. Each user in the train/test set is considered as a query, and other users in the set are considered as documents.

In a pairwise learning-to-rank algorithm [1], we accumulate cost only between pairs of items in the same list. The

Method	Precision
Layer 1	
xgboost	0.927
neural network	0.911
random forest	0.904
Layer 2	
xgboost	0.932

Table 2: Performance of classification methods on our validation set

cost L_{ij} associated with the assigning scores s_i and s_j to examples i, j respectively is:

$$L_{ij} = \log(1 + \exp(-f_{ij}(s_i - s_j)))$$

where $f_{ij} = 1$ when $y_i > y_j$, $f_{ij} = -1$ when $y_i < y_j$ and $f_{ij} = 0$ when $y_i = y_j$.

Minimizing the cost on the training data, and applying to validation/testing set gives us a ranking of each userIDs given a userID with a associated scores.

To generate the final predictions, we propose a greedy graph-based algorithm based on the local ranking scores and additional information

- Every userIDs in the test set connected with some others userIDs
- The connectivity distribution of userIDs in the train set

4.4 Implementation

The scripts to generate negative examples and prediction pairs of userIDs based on the number of shared facts/events as described in Section 3 are implemented using python[4]. In addition, we made use of several libraries including scikit-learn[5], keras[6], and xgboost[7] for learning algorithms.

5. RESULT

Table 2 presents the results of classification method on the our validation set. In the set, there is 167,055 positive examples, thus the results reported in Table 2 are computed at top 167,055. It can be seen that xgboost obtains the highest scores, while neural network and random forest get slightly lower results. Table 2 also shows the effectiveness of stacking/ensemble method, where the xgboost in the layer 2 outperforms all algorithms in the layer 1.

Table 3 shows the results of classification method on the validation leaderboard. It shows a similar spirit to the results on our validation set, in which stacking method outperforms the other algorithms.

We developed a learning-to-rank method after the validation phase is closed, thus we only report results in the testing phase of competition which are shown in Table 4.

6. CONCLUSIONS

We have showed that the learning-to-rank method outperforms the classification methods in the cross-device entity linking problem. In addition, the stacking method performs slightly better than the individual classification method. Our results can be improved by using a better algorithms to combine the ranking scores. In addition, tuning the number of prediction pairs can get higher F1 scores.

[4]https://github.com/namkhanhtran/cikm-cup-2016-cross-device

[5]http://scikit-learn.org

[6]https://keras.io/

[7]https://github.com/dmlc/xgboost

Method	Precision	Recall
Layer 1		
xgboost	0.208	0.416
neural network	0.195	0.390
random forest	0.172	0.343
Layer 2		
xgboost	0.209	0.417

Table 3: Performance of classification methods on the validation leaderboard

Method	F1	Precision	Recall
Rank1	0.4203854111	0.3987583333	0.4444929542
Rank2	0.4166958839	0.3944475328	0.4416040426
Rank3	0.4137067007	0.4004224581	0.4279026130
Rank4	0.4016845811	0.3659128735	0.4452082153
Ours (clf)	0.2922900669	0.2192172108	0.4384364579
Ours (rank)	0.3611029814	0.3322743310	0.3954093244

Table 4: CIKM Cup 2016 testing leaderboard

7. REFERENCES

[1] C. J. Burges. From ranknet to lambdarank to lambdamart: An overview. Technical report, 2010.