Cross-Device Matching for Online Advertising with Markov Cluster Algorithm at CIKM Cup 2016

Ivan Bendyna
Yandex
bendyna.ivan@gmail.com

ABSTRACT

This paper explains solution of the Cross-Device Entity Linking Challenge, which was a part of CIKM Cup 2016 (Conference on Information and Knowledge Management). The goal of the challenge is to cluster devices by browser logs - to predict that few devices belong to one user.

Keywords

Cross-Device Entity Linking Challenge, CIKM Cup, Markov Cluster Algorithm

1. INTRODUCTION

Online advertising is, perhaps, the most successful business model for the Internet known to date and the major element of the online ecosystem. Advertising companies help their clients market products and services to the right audiences of online users. In doing so, advertising companies collect a lot of user generated data, e.g. browsing logs and ad clicks, perform sophisticated user profiling, and compute the similarity of ads to user profiles. User identity plays the essential role in the success of an online advertising company/platform.

As the number and variety of different devices increases, the online user activity becomes highly fragmented. People check their mobile phones on the go, do their main work on laptops, and read documents on tablets. Unless a service supports persistent user identities (e.g. Facebook Login), the same user on different devices is viewed independently. Rather than doing modeling at the user level, online advertising companies have to deal with weak user identities at the level of devices. Moreover, even the same device could be shared by many users, e.g. both kids and parents sharing a computer at home. Therefore, building accurate user identity becomes a very difficult and important problem for advertising companies. The crucial task in this process is finding the same user across multiple devices and integrating her/his digital traces together to perform more accurate profiling.

To encourage research in this area CIKM organized Cross-Device Entity Linking Challenge[1]. Dataset provided by Data-Centric Alliance[2].

2. CHALLENGE DESCRIPTION

[1]https://competitions.codalab.org/competitions/11171
[2]Data-Centric Alliance http://datacentric.ru/en/

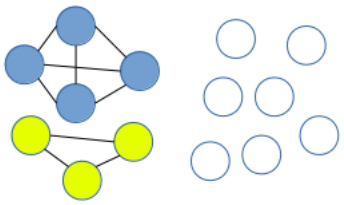

Figure 1: Train and test parts of dataset.

2.1 Dataset

DCA provided fully anonymized browser logs for a set of devices. A browsing log containing a list of events for a specific device. Each event includes URL (hashed), title (hashed), timestamp.

Example:

- URL: ed95a9a5be30e4c8/5162fc6a223f248d/ 31a42ef13edf7d8a/c2ee3aa455c8b288/ 656e9a1b3fa15996
- title: b687bc71fc33713b 19475eb9e8506115 1c202a1bece8798 0e415cc3f3ea206 f27b5d716eb58ac 37a20a398fae482a
- timestamp: 1464769462076

The hashing is done by replacing all words in the URL and title with their hash-codes. To hash URLs and titles, organizers:

1. build the vocabulary by concatenating all available textual data such as URLs and titles;
2. for each unique word, assign a hash-code using an MD5-based hash function;
3. replace each word with the corresponding hash-code.

URL words separated by "/" and "?". Title words separated by spaces.

For example, if we have 2 URLs:

- google.com/search?text=cats
- google.com/search?text=dogs

First 2 URL words are equal, so their hashes are equal too. For example:

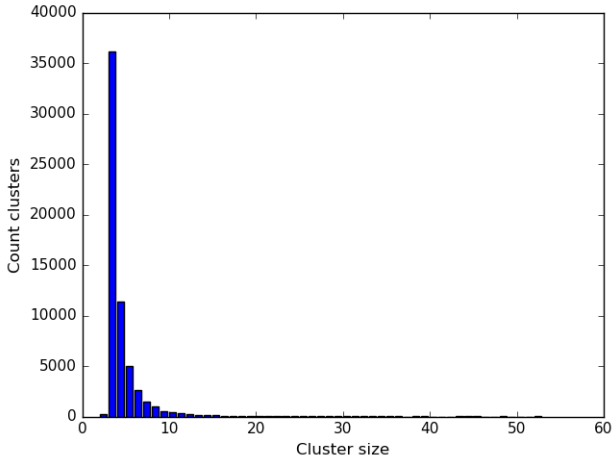

Figure 2: Count clusters of each size.

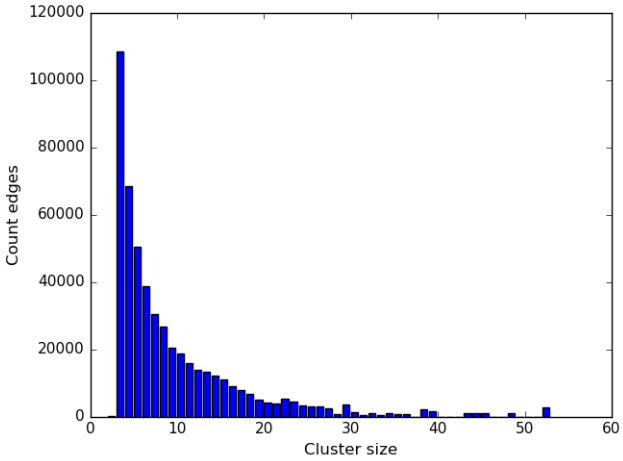

Figure 3: Count edges in clusters of each size.

- 123456/abcdef?10ab10
- 123456/abcdef?23cd23

Also, organizers provided edges for the train part of dataset. Participants should predict edges in the test part.

Here is statistics provided by organizers, so, we know count of correct edges in test part. It's interesting, but not very useful for my solution.

- The number of unique tokens in titles (dictionary size): 8,485,859
- The number of unique tokens in URLs (dictionary size): 27,398,114
- The average number of events/facts per userID: 197
- The median number of events/facts per userID: 106
- The number of unique domain names: 282,613
- The number of all events for all users combined: 66,808,490
- The number of unique userIDs: 339,405
- The number of unique websites (domina + URL path): 14,148,535
- The number of users in the train set: 240,732
- The number of users in the test set (public and private leaderboard combined): 98,255
- Known matching pairs for training: 506,136
- Known matching pairs for testing (public and private leaderboard combined): 215,307

2.2 Metric

Predicted edges were evaluated by F_1 score (a harmonic mean of Precision and Recall).

$$F_1 = \frac{2pr}{p+r}$$

But there were two phases in contest. 50% of the correct pairs were used for the first phase and 50% were used for the second phase during the final stage of the Challenge. So, in each phase half of submitted edges is useless. If we have precision p in validation set, in test set (leaderboard) it will be only $p/2$.

$$F_1^* = \frac{0.5pr}{0.5p+r}$$

It is formula of calculating leaderboard F_1^* score by real precision and recall.

2.3 Cluster sizes

We can see on Figure 2, that there very few clusters contain more than 15 devices. But the more devices in cluster, the more edges ($O(N^2)$). And because of metrics calculates F_1^* for edges, we should consider importance of cluster sizes by count of edges in clusters of each size (Figure 3).

2.4 Baseline

Organizers also provided baseline solution. It includes these steps:

- split urls into words
- build matrix devices-words
- process matrix with TF-IDF
- for each device select 15 nearest neighbours
- sort all these pairs by distance
- submit top-215307 best pairs

Leaderboard score is **0.056**.

Table 1: Recall and size (in millions)

M	K=1	K=2	K=3	K=4
2	0.1727 (0.9m)	0.0878 (0.15m)	0.0537 (0.07m)	0.0368 (0.04m)
3	0.2405 (1.8m)	0.1367 (0.32m)	0.0904 (0.14m)	0.0647 (0.08m)
5	0.3105 (3.5m)	0.1884 (0.65m)	0.1291 (0.29m)	0.0947 (0.17m)
7	0.3536 (5m)	0.2210 (0.9m)	0.1538 (0.43m)	0.1138 (0.26m)
10	0.3964 (7.3m)	0.2543 (1.4m)	0.1795 (0.64m)	0.1342 (0.38m)
20	0.4769 (14m)	0.3221 (2.9m)	0.2341 (1.3m)	0.1785 (0.79m)
30	0.5181 (21m)	0.3587 (4.5m)	0.2658 (2m)	0.2053 (1.2m)
50	0.5634 (34m)	0.4035 (7.7m)	0.3042 (3.4m)	0.2375 (2m)
70	0.5873 (46m)	0.4260 (10m)	0.3249 (4.6m)	0.2561 (2.7m)
100		0.4520 (14m)	0.3487 (6.3m)	0.2769 (3.7m)
150			0.3763 (9m)	

3. THE APPROACH

3.1 Validation dataset

Count of submissions was limited by 15/day. So, we decided to create validation dataset to check my ideas without submission. To create it, we retrieved all clusters from train dataset and selected for validation dataset random set of clusters, which has count of edges approximately equal to count of edges in test dataset. Rest clusters included in train dataset.

3.2 Workflow

1. select sample, which contains not many edges (<50 mlns), but with big recall
2. feature extraction
3. learn model and predict probabilities for validation and test dataset
4. divide top-N probabilities into complete subgraphs with MCL (pick N and parameters of MCL in validation dataset)

3.3 Sample with big recall

We want to reduce problem to classic machine learning problem i.e. to objects-features matrix, where objects are edges. But there are too many possible edges, for example in train dataset:

$$\frac{339405 \cdot 339404}{2} \approx 5 \cdot 10^{10}$$

So, we want select subset, which has big recall and is small enough, that we can train model in a reasonable time.

Consider only rare events, which can be found only at $\leqslant M$ devices. Then consider pairs of edges, which have at least K common rare events.

Recall and sizes of such subsets for different K and M are in Table 1.

For first iteration we selected samples (K=1, M=30), (K=2, M=50), (K=3, M=50) and all pairs from baseline for 30 nearest neighbours. Union of these sets has recall 0.62 on train and validation datasets and recall 0.59 on test dataset.

3.4 Feature extraction

For each pair I have created these features:

- distance in nearest neighbours
- for 1st vertex order of 2nd among all nearest neighbours (by distance)
- for 2nd vertex order of 1st among all nearest neighbours (by distance)
- for each K in Table 1, least M such that pair appear in sample (4 features)
- URLs: count common with repetition, count commot unique, count common URLs that belong to only 2 devices, count common URLs that belong to only 3 devices, count common URLs that belong to only 4 devices, count common URLs that belong to only 5 devices, count common URLs that belong to only 6-10 devices (7 features)
- same 7 features for URL words
- same 7 features for URL domain
- same 7 features for URL domain + second word
- same 7 features for titles
- same 7 features for title words
- same 7 features for title bigrams

It's 56 features total.

3.5 Model training

For training I used Random Forest and selected max tree depth on validation dataset. Best max tree depth is 30.

ROC AUC score on validation dataset is 0.957.

Leaderboard score is **0.3584**.

3.6 Sample with big recall - iteration 2

Now, we have non-full graph, where each edge has probability. Instead of 4th item in workflow we decided complete connected component of graph and add resulting edges to sample in item 1; then repeat feature extraction and training. First of all, we should limit edges of graph, because it's one big connected component. Algorithm of build limited graph is:

- sort edges by probability in descending order
- add each edge to graph in this order except 2 cases
  1. do not add edge, if its probability is less than p

Table 2: Results for various pe, pi, p, T on validation set

p, T	pe = 2, pi = 2	pe = 2, pi = 2.2	pe = 2, pi = 2.4
p=0.07, T=30	0.42310	0.42268	0.42116
p=0.10, T=30	0.42337	0.42298	0.42170
p=0.13, T=30	0.42323	0.42301	0.42177
p=0.07, T=40	0.42551	0.42505	0.42416
p=0.10, T=40	**0.42587**	0.42545	0.42476
p=0.13, T=40	0.42569	0.42541	0.42478
p=0.07, T=50	0.42520	0.42523	0.42455
p=0.10, T=50	0.42552	0.42574	0.42525
p=0.13, T=50	0.42536	0.42571	0.42525

2. do not add edge, if it unite components and product of sizes of these components is more than T

Selecting best p, T on validation dataset, we have best $p = 0$(so, it's useless, because all probabilities $\geqslant 0$), $T = 400$. Recall of these edges is 0.77 on validation dataset.

For second iteration we selected all these edges from complete subgraphs, samples (K=1, M=70), (K=2, M=100), (K=3, M=150), (K=4, M=100) from Table 1 and all pairs from baseline for 200 nearest neighbours.

Using the same features and model we have leaderboard score **0.3702**.

3.7 MCL

The MCL algorithm based on simulation of random walks within a graph. It is iterative algorithm with 3 main steps: column normalization, expansion, inflation.

Column normalization is needed to make graph matrix a column stochastic matrix (matrix elements are probabilities, sum in each column is 1).

Expansion is the step, that simulates random walks within a graph, it's only taking the power of a stochastic matrix.

Inflation changes the probabilities after random walks by taking the power of each element. It makes big probabilities bigger and small probabilities smaller.

Full algorithm of clustering includes 2 steps:

- split graph into connected components (like in previous section)
- apply MCL to each component

This algorithm has 4 parameters: p, T, power of inflation (pi), power of expansion(pe). Then we select these parameters on validation dataset. In Table 2 are results for some sets of parameters.

Best parameters are $pe = 2, pi = 2, 0 = 0.10, T = 40$.

Leaderboard score is **0.418**.

Private leaderboard score is **0.401** (4th place).

4. REFERENCES

[1] Challenge page. https://competitions.codalab.org/competitions/11171

[2] Random Forest in sklearn. http://scikit-learn.org/stable/modules/ensemble.html

[3] Markov Clustering Algorithm. http://www.micans.org/mcl/