# ADVERTISEMENT CLICK PREDICTION

*—— A Kaggle challenge*

*Team Members: Weichuan Wu, Jingjing Feng, Yiwei Sun, Yunyi Zhang*

# PORJECT DESCRIPTION:

**Can you predict which recommended content each user will click?**

Currently, Outbrain pairs relevant content with curious readers in about 250 billion personalized recommendations every month across many thousands of sites.

In this competition, We are challenged to predict which pieces of content that its users are likely to click on. Improving Outbrain's recommendation algorithm will mean more users uncover stories that satisfy their individual tastes.

The dataset for this challenge contains a sample of users' page views and clicks, as observed on multiple publisher sites in the United States between 14-June-2016 and 28-June-2016.

Each viewed page or clicked recommendation is further accompanied by some semantic attributes of those documents.

The dataset contains numerous sets of content recommendations served to a specific user in a specific context. Each context (i.e. a set of recommendations) is given a display_id.
In each such set, the user has clicked on at least one recommendation. The identities of the clicked recommendations in the test set are not revealed.

The task is to rank the recommendations in each group by decreasing predicted likelihood of being clicked.

# DATA WE HAVE

| File Name | Available Formats |
|---|---|
| documents_categories.csv | .zip (32.34 mb) |
| clicks_test.csv | .zip (135.43 mb) |
| documents_meta.csv | .zip (15.51 mb) |
| documents_entities.csv | .zip (125.67 mb) |
| promoted_content.csv | .zip (2.52 mb) |
| sample_submission.csv | .zip (99.57 mb) |
| documents_topics.csv | .zip (120.91 mb) |
| clicks_train.csv | .zip (389.75 mb) |
| events.csv | .zip (477.74 mb) |
| page_views.csv | .zip (29.71 gb) |
| page_views_sample.csv | .zip (148.51 mb) |

# TRAIN DATASET AND TEST DATASET:

```
> click_train <- fread("clicks_train.csv")
Read 87141731 rows and 3 (of 3) columns from 1.385 GB file in 00:00:14
> head(click_train)
   display_id  ad_id clicked
1:          1  42337       0
2:          1 139684       0
3:          1 144739       1
4:          1 156824       0
5:          1 279295       0
6:          1 296965       0
```

```
> click_test <- fread("clicks_test.csv")
Read 32225162 rows and 2 (of 2) columns from 0.472 GB file in 00:00:04
> sample_submission <-fread("sample_submission.csv")
Read 6245533 rows and 2 (of 2) columns from 0.254 GB file in 00:00:23
> head(click_test)
   display_id  ad_id
1:   16874594  66758
2:   16874594 150083
3:   16874594 162754
4:   16874594 170392
5:   16874594 172888
6:   16874594 180797
> head(sample_submission)
   display_id                                                ad_id
1:   16874594                66758 150083 162754 170392 172888 180797
2:   16874595                                         8846 30609 143982
3:   16874596   11430 57197 132820 153260 173005 288385 289122 289915
4:   16874597 137858 143981 155945 180965 182039 285834 305790 308836
5:   16874598                           67292 145937 250082 335632
6:   16874599                91681 163776 173130 210516 213116 296295
```

# HIGHLIGHT —— A BIG DATA PROJECT AND OUR SOLUTION

- **Main Difficulties**:
- 1. It's our first time being faced with such **a large dataset** which **couldn't be handled by PC**.
- 2. Most of the **variables are categorical**, which means we couldn't just directly use  predictive models.
- 3.When doing calculations, we have to be very careful to make sure we have enough memory.
- So, **we want to highlight our methods to handle big data step by step.**

# DETAILS

- 1. read data:
- we can't use page_view.csv(>100 GB), so we decide to use other relational dataset and combine them together.
- We tried SQL but it took more than 8 hours to joint those tables. Then we used the "match" function in R (15 minutes in total) to get the merged train dataset(# of row>80000000)

| ad_id | clicked | document_id | source_id | publisher_id | category_id | topic_id | ad_document_id | campaign_id | advertiser_id |
|---|---|---|---|---|---|---|---|---|---|
| 42337 | 0 | 379743 | 6482 | 24 | 1203 | 285 | 938164 | 5969 | 1499 |
| 139684 | 0 | 379743 | 6482 | 24 | 1203 | 285 | 1085937 | 17527 | 2563 |
| 144739 | 1 | 379743 | 6482 | 24 | 1203 | 285 | 1337362 | 18488 | 2909 |
| 156824 | 0 | 379743 | 6482 | 24 | 1203 | 285 | 992370 | 7283 | 1919 |
| 279295 | 0 | 379743 | 6482 | 24 | 1203 | 285 | 1670176 | 27524 | 1820 |
| 296965 | 0 | 379743 | 6482 | 24 | 1203 | 285 | 1133290 | 28334 | 3922 |
| … | … | … | … | … | … | … | … | … | … |

# THE PROGRESS OF RE-PRECESSING DATA

click_train: display_id
ad_id
click_status

*joint with event.csv*

*using the same display_id*

click_train: display_id
document_id
ad_id
click_status

*joint with 4 doc csv*

*using the same document_id*

click_train: display_id
document_id
source_id
topic_id
categories_id
publisher_id
ad_id
click_status

*joint with promoted_content.csv*

*using the same ad_id*

***final:*** display_id
document_id
source_id
topic_id
categories_id
publisher_id
ad_id
compaign_id
advertiser_id
click_status

# DETAILS

- 2. Using features of advertisement and document:
- all the features are categorical, and we couldn't find a practical method to map features to features.
- logistic and multinomial regression didn't work. Too many categories of the response(probability of ad_id given feature, # of unique ID is 478950) so it's hard to implement those algorithms. (not enough memory, even computers in computer lab)
- So we calculated the frequency to predict test data, while during this progress, we dropped loops and used matrix. (no more than 2 hours)

# HOW TO CALCULATE PROBABILITY

$$P(\text{ad feature} \mid \text{doc feature}) = \frac{\text{The number of click of the ad feature given doc feature}}{\text{The number of recommendation of the ad feature given doc feature}}$$

- **Ad feature**: ad_id, advertiser_id, campaign_id

- **Doc feature**: topic_id, categories_id, publisher_id

- Then we can calculate the probability matrix based on features of advertisement and document.
- And there are several combinations. i.e. p( ad_id | topic_id )

# OUR RESULTS AND THE COMPARISON

## Scores we achieved in Kaggle

| Doc \ Ad | Ad_id | advertiser_id | campaign_id |
|---|---|---|---|
| topic_id | 0.60187 | ———— | ———— |
| categories_id | 0.63269<br>rank 388 | 0.62657 | 0.63178 |
| publisher_id | 0.63689<br>Top 240 | **0.6397**<br>**top 100** | **0.64158**<br>**Top 17%** |

# LIMITS AND IMPROVEMENTS

- 1. We couldn't find practical algorithms to deal with all the categorical data, which made our prediction was based on frequency in stead of using algorithm to map features to features
- 2. There are limitation about our PC so we couldn't use page_view.csv(>100 GB) which contains the most information.

- IMPROVEMENTS: finding a way to deal with categorical data and using cloud computing platform to analyze page_view.csv