Outbrain click prediction team

Dean Ayalon, Yair Hadas, Eran horowitz

Documentation

## Business Problem to Machine Learning Problem

Companies like Outbrain are in the business of online advertising. Many high-profile websites include advertising based on Outbrain's platform, and many companies choose Outbrain as their online advertising platform. Both sides' goal is the same - to maximize number of times that ads are being clicked. Every click means another potential customer or potential revenue for the advertiser, which pays for Outbrain in return. This problem requires an ability to forecast which ad is most likely to be clicked. But how?

This is where Machine Learning comes in. Using the vast amounts of data presented by Outbrain as part of this challenge, we can build a model that would attempt to accurately predict which ad a given user will click presented a given series of ads. The info from this model could be used to design a better algorithm for choosing which ads will be presented to each user.

## A Formal Definition of the Problem

The Outbrain click prediction contest is fairly simple: groups should come up with models that use data for a given document and all the ads that appear on that document (roughly 3-12 ads, that form together a display), and predict which of the ads will be clicked. As the ads are themselves documents, one can compare the document and the ad in many ways: their publisher, publish time, topic, etc. Using those connections, we have come up with almost 20 features, but only about half of them were found useful for prediction. This provides some meaningful insights about the nature of ads that show in documents; many guesses that we had had regarding the nature of clicks turned out to be false. Below can be found a list of features, each detailed and commented upon, if found relevant or not. Our prediction result is roughly equal to one of the 50 best teams (out of almost 1000 teams).

## Introducing the Data

Outbrain provided two weeks’ worth of data for this challenge, for the period of 14-28.6.2016, containing information about displays, the sites they were displayed on, the ads they contained, various details about the documents these ads were linked to, the users viewing those ads, and so forth.

The full dataset for this challenge was enormous, totaling almost 100 GB when uncompressed. As we lacked the resources the handle such vast amounts of information, we chose to focus on only a small subset of the data. The full training file contains information about over 16 million different displays, we chose to sample about 3% of that amount, so the data set we worked with contained about 500,000 displays, uniformly sampled from the original data.

Some of the tables that we worked with:

**clicks\_train.csv** - the core table, containing display id's, the ads each display contains, and which of them was clicked.

**events.csv** - contains information about the clicks "events" - on what platform they were made (mobile, desktop or tablet), the country from which they were made, when they were made, etc.

**documents\_categories.csv, documents\_topic.csv, documents\_entities.csv, documents\_meta.csv** - tables containing various metadata details about documents, both ones which ads were displayed on, and ones which were linked to by ads.

**promoted\_content.csv** - contains information about the publishers of ads and the campaigns they were part of.

The biggest table in the data was **page\_views.csv**, which was essentially the same as events.csv, but contained **all** of the page views of users tracked in those two weeks, including pages they viewed without clicking an ad. We decided to abandon this table due to the massive resources we would have needed in order to work with it.

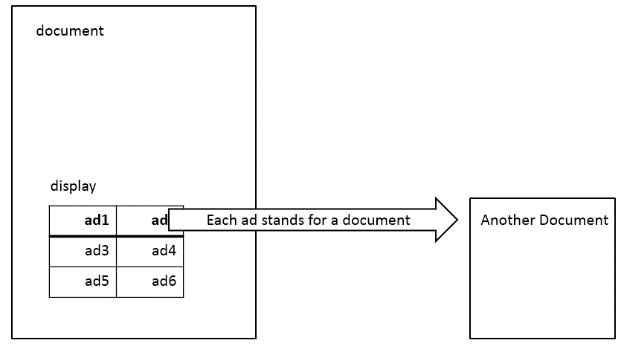
## Data Preparation

One of the main challenges we faced was deciding how to integrate the many tables contained in the data set to one comprehensive table on which we can run our models. It took us a while to understand the connections between them, and the meanings of the various features contained in each of them. Some of the columns had plenty of null values (e.g. publish\_time, publisher\_id), so instead of using them we ended up relying only on columns which had no or very few null values.

The few null values we did need to impute/clean are explained individually for each relevant feature.

## Features Extraction

The Outbrain competition requires us to estimate the probability that an ad be clicked, given a set of other ads that are shown along with it, which form together a *Display*. That is, the basic tuple for the final features table, on which we will run the Machine-Learning algorithm, is (doc, ad). That is, all features will be descriptive of that basic unit. For instance, the *Topics* attribute, which is detailed in the next section, is a measure of the topic-similarity between doc and ad. A click on a specific ad is an *event*.



##### 0.b. Simplifying CHOICES

The most important simplification we have made is to ignore user attributes, due to two important factors. First of all, the file containing most information about the users is the *page\_views.csv* document, which weighs more than 100 GB. The sample provided by the Outbrain team contains a very little portion (less than 1%) of all of the page views, and does not correspond to the other tables. That means that most of the actual event we worked on were not found in the page\_views sample. The second reason is that, when reading entries in the forum, we found that the users have not played a major role in winning models.

Also, we chose to sample only about 3% of the displays contained in clicks\_train.csv. This was still a massive dataset (over 500,000) displays, but much more manageable in terms of time required to do calculations. The sampling was uniform without any filtering or conditions.

After choosing the displays we worked with, we filtered all of the tables to include only entries relevant to those displays and the documents they referred to.

##### 1-3. find\_similarity: Categories, Entities, Topics

**1-3.a. Overall Review**

Outbrain has provided three different parallel documents, containing direct relations between documents and their corresponding Categories, Entities and Topics. All those elements are serialized to id's – one does not know what the topic with id 200, say, stands for. Each document has usually more than one of each element, mostly between two to six, and every element has a corresponding confidence level. Confidence levels for a document don't accumulate to a given value (e.g., 1), but can range between very low overall confidence levels – sum of all is less than 0.1 – or slightly higher than 1.

Although the tables are completely parallel, there are major differences between them. First of all, in the Entities table there are about as many entities as there are documents; that means that the chance for two different documents containing the same entity is very low. But the situation in the two other tables is much better: there are 300 topics and about 2000 categories.

**1-3.b. Feature Extraction**

The straightforward guess, of course, is that two documents containing the same element – entity, category or topic – will be more prone to be clicked from one another. This requires some confidence-sharing formula. The initial one we came up with was based on the 1st element from each. Namely, the value that goes in the feature is 1 if the highest confidence elements are the same. Otherwise, they value to 0. Formally,

But there is high variance in the confidence vectors, and if we used that function, it would result in equivalence of a very low confidence and a very high one, when both are highest per document. Therefore we came up with a more sophisticated equation, which takes into account all elements which the two documents share,

Where confidence1(i) and confidence2(i) and are the confidence measures of the *ith* shared element(topic, category or entity) in documents 1 and 2 respectively.

##### 4-5.Ads Per Advertiser and Ads Per Campaign: advertiser\_freq and campaign\_freq

We wanted to measure the "attractiveness" of every ad - whether it's because it was published by a big-name publisher or because it mentions some high-profile person or company. As the topic and entity data that were provided are purely numeric, we couldn't do this directly.

Our solution was to check how many ads each advertiser published - working under the assumption that "attractive" advertisers would be big ones with a lot of money and high-profile customers, meaning they will have many more ads published than small publishers with small budgets. As shown in the notebook, and according to our hypothesis, more prolific advertisers and campaigns were indeed clicked on more often.

##### 6. event\_time

Every event, or *click*, occurs at a different time of day. Using four time zones for the most popular countries (which constitute well above 90% of the data, being English-speaking) – England, USA, Canada, and Australia. We chose to bin the times into 5 values: Morning - 7:00 to 11:59, Noon - 12:00 to 15:59, Afternoon - 16:00 to 19:59, # Evening - 20:00 to 23:59, Night - 00:00 to 06:59.

As the timestamps provided are all UTC, we also corrected times for those 4 countries accounting for timezone differences. The correction values were determined based on each country’s timezones, the fact the data was collected during the summer (therefore Daylight Savings Time was applied), and which parts of each country are most populated. The final correction values are as follows: UK - +1 hour, US and Canada - -5.5 hours, Australia - +10.5 hours.

##### 7. Platform

A simple binary encoding of the "platform" feature, provided in the events.csv table. Both this attribute and the next one rely on the assumption that click trends will differ between the platforms and between times of the week. Nevertheless, we still didn't know exactly how they should influence click tendencies, so we relied on Machine Learning to tell us that.

##### 8. Is\_weekend

Another simple binary feature – we thought weekend click trends might be different enough to make this feature useful, but it actually turned out to not be very helpful in the final model.

##### 9. clicks\_appearances\_ratio

Measures the ratio between the number of times an ad was clicked and how many times it has appeared. As this ratio is very diverse, we have scaled it by substitution of average and division by standard deviation. But, as this scaling produced slightly lesser accurate results, we have used the unscaled attribute. The results matched our expectations, as can be seen in the notebook. Another factor that supports the usefulness of this attribute is that using only this factor, groups have managed to reach an accuracy of 0.63714 (see appendix), which therefore served as a benchmark for us.

##### 10. ad\_count\_per\_display

Measuring how many ads are found in a given display. This feature has been found relevant for the model, supporting our hypothesis that ad click tendencies will differ along with number of other ads in the display. We assumed that this seemingly meaningless factor would prove meaningful, and we were right. And it makes sense, since one might choose to click a specific ad if it appears just with two other ads and not with ten. Here, once again, the "blind power" of Machine Learning has made things clear for us and proved this attribute meaningful.

##### attributes we have dropped

1. **topic\_popularity**

Number of times that the ad's topic was clicked. This feature is very logical, since we tend to assume that a popular topic will be clicked with a higher probability, but this feature didn't improve prediction and was dropped.

2. **publisher\_popularity**

Measures how many times ads by the ad's publisher were clicked. This was dropped was way too many rows (over 30%) had null publisher\_id values.

3. **ad\_age**

We wanted assess each documents “click age”, i.e. the time it was clicked minus the time it was published. This dropped as less than 70% of rows had valid publish\_time values, so we decided not to pursue this feature further.

4. **XCTRY**

A general method for extracting CTR features – one of the top teams included many features of this type. This method computes a simple probability measure for shared attributes. For instance, in the campaignCTRpublisher it calculates pr(click==1|campaign,publisher) using the Bayes formula. We abandoned it only due to lack of time.

Other, less significant attributes that we have dropped can be found in the GitHub directory.

## Model Evaluation

Models in the competition were evaluated using the MAP@12 metric, MAP standing for Mean Average Precision:

In this formulation, |U| is the number of display\_id's, and P(k) is the precision at cutoff k. To try and get a better understanding of this metric, we can think of the array of ads we give in a response to a display id as a **series** of predictions, trying to predict the ad that was actually clicked. Naturally, the best case would be to make the correct prediction in the first try, but making it on the second or third attempt isn't bad either. On the other hand, making it on the last attempt isn't that satisfying. The MAP metric represents this difference.

For our data, since for every display there is only one correct answer (the ad that was actually clicked) the formulation becomes a bit simpler:

In this formulation, is the position of the correct ad in the array we return. i.e. 1 if it's first, 2 if it's second, etc.

## Model Selection

The “immediate suspect” for a model that would help us analyze this dataset was logistic regression, as this model is designed specifically for data where the label is binary. Also, at first we really weren’t aware of any other models which fitted this framework.

So most of our initial work was done using logistic regression. Later, when we became aware of numerous other models that could be fitted to this data, and especially of how easy it is to examine different models using the sklearn library, we started investigating other models.

At first, to select the best candidates for parameter tuning, we examined several different models using their default parameter values. Many competition entries used some form of logistic regression, but also XGBoost, which is an implementation of the gradient boosting algorithm. In order to have more options, we also decided to test other ensemble methods, namely Random Forest and AdaBoost. We first ran all algorithms with default hyper-parameters. The results were as follows:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Model** | Logistic Regression | Gradient Boosting | Random Forest | AdaBoost |
| **MAP@12 Score on test data** | 0.67921 | 0.68042 | 0.62038 | 0.67853 |
| **Zero-One Score on test data** | 0.48338 | 0.48452 | 0.40908 | 0.48198 |

Random forest is clearly lacking in performance compared to the other models, so we decided not to bother with optimizing it. All other models, however, are quite close to one another in terms of accuracy scores achieved, Gradient Boosting narrowly beating the other two. In the notebook we included only the top two models, Logistic Regression and Gradient Boosting, and applied parameter tuning to both of them to determine which of them is best.

## In conclusion

In this project we have provided the best prediction we could for the chance of an ad in a display to be clicked. We discovered that the strongest feature required for the prediction is the obvious one - the simple relation between the number of time an ad appeared, and the number of times it got clicked. Yet the hard work, as we learned from the discussions in Kaggle (and also through our own experience), is to improve the accuracy even more - and in order to do that we were required to think hard on meaningful features to extract from the data, and also to choose the most accurate model to predict with. It is important to state that this competition is much more about accuracy of prediction than about meaningful insights on data; and indeed, in some features such as the *is\_weekend* attribute, we have relied much on the "blind force" of our algorithm to match our intuition. This indeed was the case with the winning teams as well: what won the competition were strong attributes and strong models.

**Appendix**

External Sources we used:

Posts on the Kaggle Forum:

Top Teams Solutions:

<https://www.kaggle.com/c/outbrain-click-prediction/discussion/27977>

<https://www.kaggle.com/c/outbrain-click-prediction/discussion/27923>

<https://www.kaggle.com/c/outbrain-click-prediction/discussion/27926>

Benchmark model using only CTR and reaching score of 0.63714:

<https://www.kaggle.com/clustifier/outbrain-click-prediction/pandas-is-cool-lb-0-63714/code>

Article supplied by Amit regarding CTR:

[http://www.csie.ntu.edu.tw/~cjlin/papers/ffm.pdf](http://www.csie.ntu.edu.tw/%7Ecjlin/papers/ffm.pdf)