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

In the Outbrain click prediction contest we were asked to find the best model that predicts which ad from a display has the best chance to get clicked, and for every display order those ads in a descending order of their respective click probabilities. A display is a set of ads.

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 and various details about them and about the documents they were linked to. 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 (which are almost 3 million ads), uniformly sampled from the original data.

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 (see appendix). We decided to abandon this table due to the massive resources we would have needed in order to work with it (see appendix for more details). For a better understanding of the table please view our Jupyter notebook.

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. Though in most of the tables there are no null values, some of the data was not relevant most of the ads (publishers for example) and that has been discovered only after a merge between tables. Some displays that had missing values or values that didn't make sense (like an ad with the platform value '//N') we discarded, since there were very few displays like this. In another case we filled the null with the average value of the feature (is shown in the notebook).

FEATURES EXTRACTION

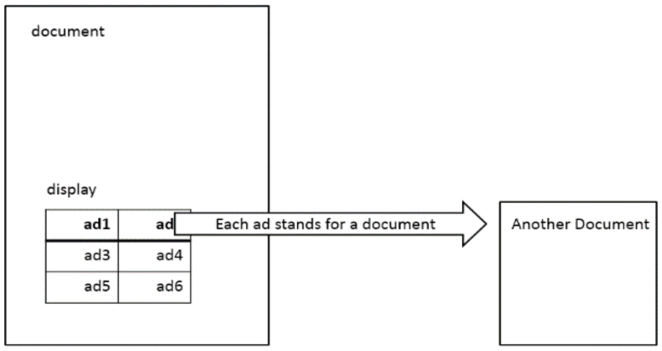
0. SIMPLIFYING CHOICES

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

We asked ourselves if users tend to click on an ad that has a topic related to the topic of the document it appeared in. as can be seen our wonderful illustration (to the right), each ad appears in a document and leads to a document.



To check this idea we came with the following formula, in order to calculate the similarity value of every element in the relevant tables (which are topic, category and entity):

$$shared\ confidence = \sum_{shared\ element\ i} confidence1(i) * confidence2(i)$$

Where confidence1(i) and confidence2(i) and are the confidence measures of the i^{th} shared element 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 click event occurs at a different time of day. The timestamps provided only counted ms from the first click event in the data, to get to real timestamps we added 1465876799998 (number of ms from 1.1.1970 to first click) to all of them. We also decided to work in seconds instead of ms, as this made it easier to work with time related functions. As the timestamps provided are all UTC, we also applied timezone corrections to the four most common countries in the data – 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. 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. This turned out to be an important feature in the final model.

11. TOPIC_POPULARITY_CONF

This features tries to measure the popularity of each topic by measuring the amount of clicks they got divided by the number documents associated with it. As each document relates to several topics, this is calculated for each of them and then summed, each popularity figure multiplied by the respective topic's confidence.

12. COUNTRIES

In order to give more weight to the geographic element of the event, we added four binary vector, one for each major country (US, Canada, UK and Australia), to the final dataset.

FEATURES WE DROPPED

1. **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.

2. **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.

3. **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 discarded this features because we got memory errors while using it – the pandas library couldn't work even with the relatively small sample of the data we had, and other solutions increased the running time significantly.

4. **count_states**

We wanted to add a vector which counts the appearance of each state. It was discarded due a large amount of nulls (25%).

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:

$$MAP@12 = \frac{1}{|U|}\sum_{u=1}^{|U|} \sum_{k=1}^{min(12,n)} P(k)$$

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:

$$MAP@12 = \frac{1}{|U|}\sum_{u=1}^{|U|}\frac{1}{r_u}$$

In this formulation, r_u 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 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.

All non-boolean features were scaled to zero mean and unit variance as a basic pre-proccessing step.

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.

Random forest was 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. The parameters to tune were selected according to sklearn's documentation (further explanation about the tuning is in the notebook since it goes better with the work flow).

IN CONCLUSION

In this project we attempted to provide the best prediction we could for the probability 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. 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:

The Outbrain competition in Kaggle:

https://www.kaggle.com/c/outbrain-click-prediction

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

A kernel posted on Kaggle showing that events.csv is practically a subset of page views.csv:

https://www.kaggle.com/gspmoreira/outbrain-click-prediction/unveiling-page-views-csv-with-pyspark