Outbrain Click Prediction

Piyush Shinde[1], Rahul Raghatate[1], Saurabh Kumar[1]

Abstract
"Click Here!". Isn't it fascinating how ad clicks can generate tons of revenue? To boost the income through ad clicks, it is imperative to understand the significance of the factors affecting ad clicks. Outbrain, the web's leading content discovery platform, matches relevant content with users in about 250 billion personalized recommendations each month across many thousands of sites. In this report we try to improve Outbrain's algorithm to predict which pieces of content it's global base of users are likely to click on.
Based on the analysis of the huge dataset provided by Outbrain, our primary prediction model sorted ads according to their prior probabilities of being clicked. Our next approach included penalizing and awarding ads based on their click counts. Subsequently, we included ad attributes such as advertisers and campaigners, followed by ad display platform. In a nutshell, we did not use all the features due to our machines' inability to handle huge-datasets. We incorporated few transformed features for our click prediction model which uses Naïve Bayes and Stochastic Gradient Descent. The designed model predicts ad clicks with an accuracy of 80.98%.

Keywords
Click Through Rate — Naïve Bayes — Stochastic Gradient Descent

[1] *Data Science, School of Informatics and Computing, Indiana University, Bloomington, IN, USA*

Contents

Introduction 1
1 Background 1
2 Algorithm and Methodology 2
2.1 Input . . . 2
2.2 Early Analysis . . . 3
2.3 Preprocessing and Feature Selection . . . 4
2.4 Models . . . 5
Naïve Bayes • Stochastic Gradient Descent
3 Experiments and Results 5
3.1 Build Specifications . . . 6
4 Conclusion 6
5 Future Work 6
Acknowledgments 7
References 7

Introduction

Utilizing implicit feedback is one of the most essential techniques for promotion of content through ads and reach millions or billions of users. Implicit feedback can be regarded as a vector of attribute values, including the entities, topics, timestamps, localities, platform, advertiser etc. Given a query, whether user clicks an ad is strongly correlated with the user's opinions on an ad.

The implicit feedback is readily available these days. Terabytes of such data is produced every day, with which content management for various promoters has become easy and has led to fast adaptability to the needs of users by putting the most relevant advertisements in the most conspicuous places. Implicit feedback such as click data has been used in various ways: towards the optimization of search engine ranking functions. [1][2][3]

The above-mentioned work mostly relies on a core method: to learn a click model. The search engine logs a large number of real-time query sessions of users, along with the user's response (clicked or not) flags. This data is regarded as the training data for the click model, which will be used for click predictions for future sessions. When normalized by amount of exposure that is typically measured by the number of times an article is shown, we get a measure called click-through rate (CTR), i.e., number of clicks per display; throughout we use CTR to select the best ads-group to be displayed on the related web-pages. This prediction power can help improve the revenue for ad promotion and content management agencies, and play an essential role in search auctions (e.g [4]).

In this report, we discuss how clickthrough data is, and how it can be used to generate recommendations in the form of preferences. We tried to explore the implementation of Naïve Bayes, SVM, Random Forest and Stochastic Gradient Descent algorithms for learning parameterized orderings in Section 2. Section 3 evaluates the model based on experimental results.

1. Background

Finding the principle features which affect users' click behavior is crucial in click predictions for ads . There has been much related research in the this area. Richardson, Dominowska, and Ragno's [5] research focuses on using features of ads, terms, and advertisers to learn a logistic regression model

in order to predict clicks for new ads which will improve performance of an advertising system. In contrast, our approach is to use classification model for those ads having enough click log data and relational data w.r.t other advertisements it is being displayed with. Thus, there should be differences in developing features and building models for both new ads and ads with historical click log data [5].

In Cheng and Cantu-Paz's [6] work, user-specific and demographic-based features were developed to reflect the click behavior of individuals and groups with a maximum entropy model applied. However, it won't be possible for the Outbrain data as Outbrain has provided data logs for 14 days period which is approximately 104 Gigabytes which has extreme uniqueness in user page views and demographic data being distorted which requires lot of data preprocessing.

Kim, Qin, Liu, and Yu's [7] applied logistic function in a factor graph model to improve ad quality and advertising effectiveness, and then tried to find the factors that could explain user click behaviors. This model is similar to our approach except we were not provided with the query information. Hence, we were concentrating towards advertisement features like its advertiser, campaigner, platform on which it is being displayed (mobile, desktop, tablet) and the document attributes like its topic, category and content displayed as discussed in Section 3.

2. Algorithm and Methodology

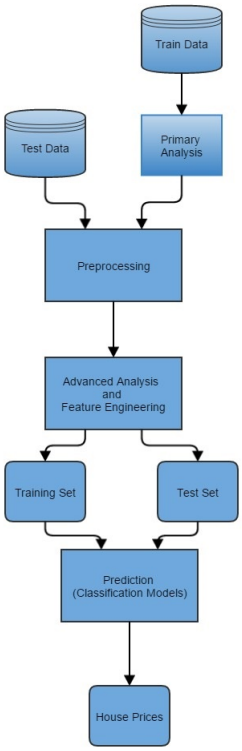

Figure 1. Architecture Diagram

2.1 Input

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.[8]

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. While most of the tables were small enough to fit in memory, the page views log (page_views.csv) is over 2 billion rows and 100GB uncompressed.

- Each user in the dataset is represented by a unique id (uuid). A person can view a document (document_id), which is simply a web page with content (e.g. a news article). On each document, a set of ads (ad_id) are displayed.

- page_views.csv is a the log of users visiting documents. To save disk space, the timestamps in the entire dataset are relative to the first time in the dataset. It includes the columns uuid, document_id, timestamp, platform (desktop = 1, mobile = 2, tablet = 3), geo_location (country > state > DMA) & traffic_source (internal = 1, search = 2, social = 3).

- clicks_train.csv is the training set, showing which of a set of ads was clicked. It includes the features display_id, ad_id & clicked (1 if clicked, 0 otherwise).

- clicks_test.csv is the same as clicks_train.csv, except it does not have the clicked ad.It contains display_ids from the entire dataset timeframe.

- events.csv provides information on the display_id context. It covers both the train and test set. It includes the features display_id, uuid, document_id, timestamp, platform & geo_location.

- promoted_content.csv provides details on the ads. It includes ad_id, document_id, campaign_id & advertiser_id.

- documents_meta.csv provides details on the documents and includes document_id, source_id (the part of the site on which the document is displayed, e.g. edition.cnn.com), publisher_id & publish_time.

- documents_topics.csv, documents_entities.csv, and documents_categories.csv all provide information about the content in a document, as well as Outbrain's confidence in each respective relationship. For example, an entity_id can represent a person, organization, or location. The rows in documents_entities.csv give the confidence that the given entity was referred to in the document.

2.2 Early Analysis

Our Firt step towards data analysis of the huge dataset was to check the consistency between the testing and training data.We checked the distribution of ad counts in both the sets, by taking the intersection between the unique ads of both the sets. The results showed a 82.87% overlap of ads between training and testing data providing enough evidence that maximum data from the testing set was present in the training data.

Next we checked the views count of each ad which gives us insight about ads frequencies ranging from less than 5 times to thousands of times. Thus, we should logarithmically transform all these values. To visualize this, we plotted the histogram for the views on each ad.

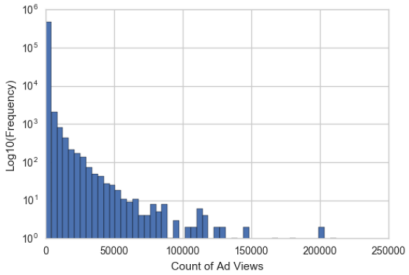

Figure 2. Count of Ad Views

As we can observe, many ads were viewed very less number of times. To check their approximate percentage, we further plotted the frequency of ads that were viewed.

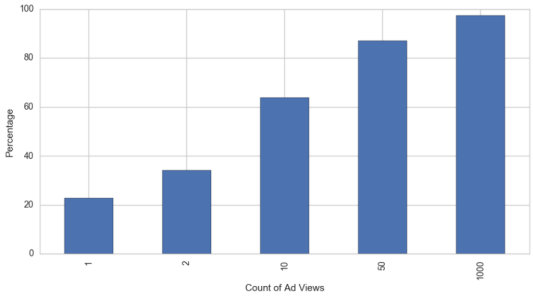

Figure 3. Percentage of Ad Views

As we can observe more than 60% of the ads were viewed less than 10 times which suggests that we must predict whether a user will click on an ad not just based on the past statistics of the ad views and clicks, but by also linking it to other ads.

We analysed the ad clicks data by plotting the histogram of the count of ads clicked. Since the results of the plotted histogram of ad views suggested many ads had a very less view count, we could expect the count of ad clicks for many ads to also be less. And the result was as expected, many ads were viewed very less number of times. To check their approximate percentage, we further plot the frequency of ads being clicked. From Figure 5, we can deduce nearly *80% of the ads were clicked less than 10 times*.

Next we analysed user clicks. We plotted the histogram for user clicks. From Figure 6, we observed that many users

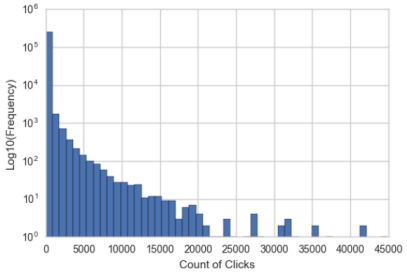

Figure 4. Count of Ad Clicks

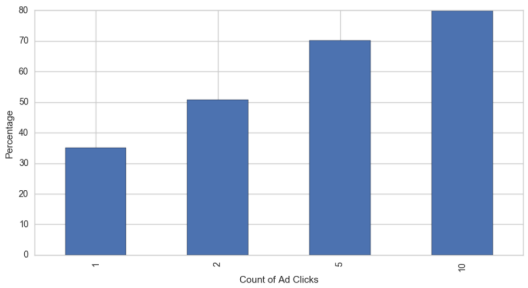

Figure 5. Percentage of Ad Clicks

clicked ads less than 5 times.

To check their approximate percentage, we plotted the frequency of user clicks. From Figure 7, we can see almost 88% of the users clicked ads just once,giving opportunity to check if users clicked the same ad more than once. The results showed around 0.01112% user only clicked the same ad more than once. These stats suggest recommending ads to users based on the user ad-click history would have very less scope.

Analysis effect of location on the number of ad clicks. From Figure 8, we interpreted that 80% of the ad-click data was from US.

Next, we investigated about how crucial is the role of platform in ad clicks. From Figure 9, we observe the ad clicks on platform 2 (mobile) were almost half of the total ad clicks on any platform.

Next, we analysed data with respect to timestamp feature. The given data is from the time frame of 14-June-2016 to 28-June-2016. On plotting the histogram of train and test data with respect to day from Figure 10 we can deduce nearly half the test data is sampled from the same timeframe as the training data, with another half sampled from the two days

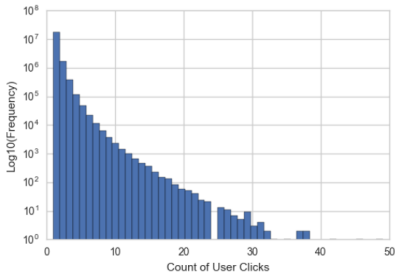

Figure 6. Count of User Views

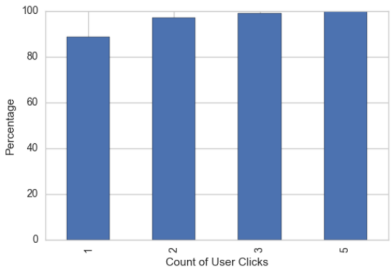

Figure 7. Percentage of User Views

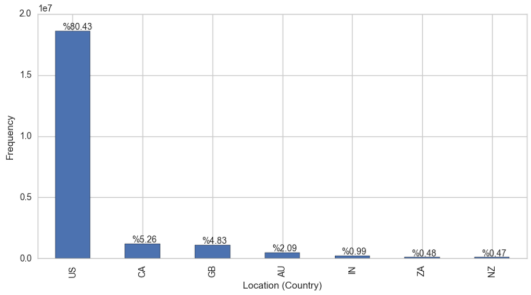

Figure 8. Country-wise percentage of Ad Clicks

immediately following. Thus, we interpret that the data has a time-based train-test split. This suggests that Outbrain wants us to predict the ad clicks for future as well as the past data (to check the accuracy of the prediction algorithm).

Analysis of document categories From Figure 11, we can see almost 80% of the categories appeared at most 15000 times. This suggests that the categories can be used for designing our prediction model.

Analysis of document entities From Figure 12, we can see almost 75% of the entities appeared just once. Since most entities appeared just once, designing the model based on entities would not affect the accuracy of the model extensively.

Analysis of document topics From Figure 13, we saw almost 47% of the topics appeared less than 1500 times. Thus,this uniqueness in topic ids occurence suggested that using this feature in designing the model would greatly affect it's prediction accuracy. Next we analysed data based on publisher and source of the document. We realized that a huge chunk of data in these features were missing. Hence, we did not use these features.

2.3 Preprocessing and Feature Selection

As we tried to incorporate all the features to create master data set associated with ads, we faced the memory challenge

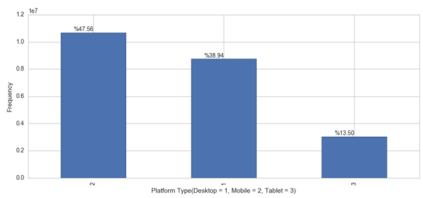

Figure 9. Platform-wise percentage of Ad Clicks

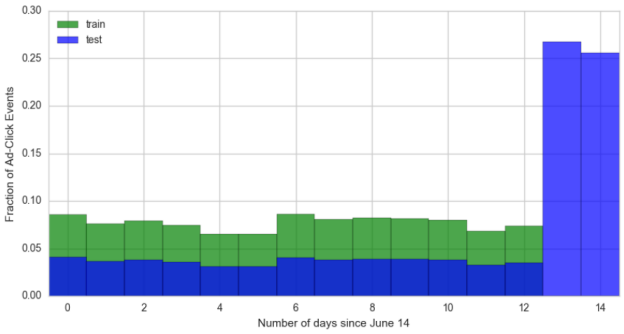

Figure 10. Test-Train data split

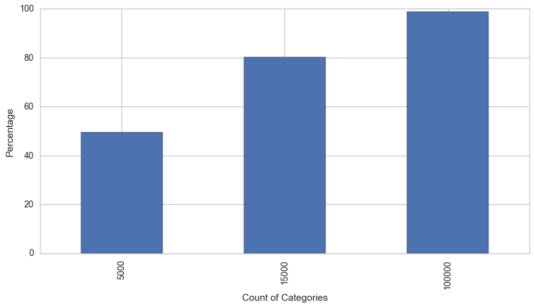

Figure 11. Count of Categories

and had to drop following principle variables:

- Ad attributes like traffic_score ,geo_location and timestamp from events
- User frequency from page_views
- Document attributes like entity, topic and category

Based on the data analysis and memory constraints for the master dataset creation, we decided to use following principle features for classification model.

- Platform from events
- Advertiser_id and Campaigner_id from promoted content

While analysing platform attribute values, we discovered that it has mixed data [string and integer values] which requires pre-processing.It contained 7 levels of platform attribute instead of main 3 levels of Desktop, Mobile & Tablet.

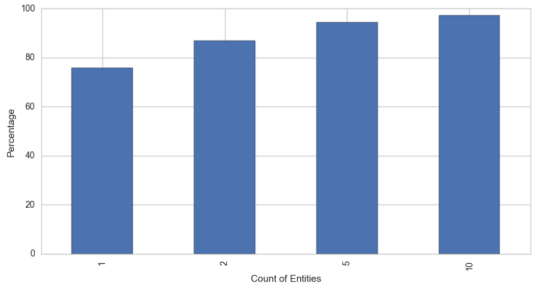

Figure 12. Count of Entities

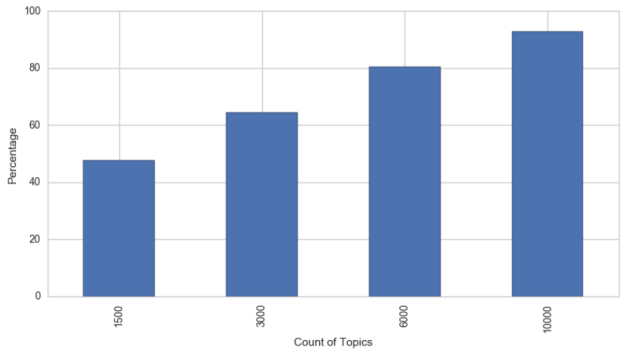

Figure 13. Count of Topics

Hence, we converted all values for the categorical variable platform from 7 levels to 3 levels. Moreover, instead of using categorical variables advertiser_id and campaigner_id ,we transformed them to nominal variables with values based on the click score of ads associated with advertiser and campaigner.

2.4 Models

2.4.1 Naïve Bayes

It is a supervised learning technique, used to predict a target function,

$$f : X \rightarrow Y$$

This is equivalent to calculating the conditional probability of P(Y|X), where X = set of features, Y = set of classes. For instance, classification of child and adult data on the basis of height and weight.
To apply Naïve Bayes:

$$P(Y = y_i|X) = \frac{P(X|Y = y_i)P(Y = y_i)}{\sum_j P(X|Y = y_j)P(Y = y_j)} \quad (1)$$

$$P(X|Y) = \prod_m P(X = x_m|Y) \quad (2)$$

where, Y = y_j.

$$P(X = x_m|y_j) = \frac{1}{\sqrt{2\pi\sigma_{y_j}^2}} exp\left(-\frac{(x_m - \mu_{y_j})^2}{2\sigma_{y_j}^2}\right) \quad (3)$$

2.4.2 Stochastic Gradient Descent

It uses supervised learning to classify convex loss functions. It computes the gradient descent over single example to perform a single feature(parameter) update. This characteristic of SGD enables it to be used for online learning such as for streaming data. The loss function is a function of features. The classifier predicts values of the loss function based on the features so as to classify the data under different labels.

Repeatedly updates θ:

$$\theta = \theta - \eta\Delta_\theta L(\theta) \quad (4)$$

SGD has been successfully applied to large-scale and sparse machine learning problems often encountered in text classification and natural language processing. Given that the data is sparse, the classifiers in this module easily scale to problems with more than 10^5 training examples and more than 10^5 features.[8]

3. Experiments and Results

We came up with several solutions to sort the ads based on their likelihood of being clicked. Predicting future click probabilites for a set of ads in the following ways:

1. Count of ad Views
2. Count of ad Clicks
3. Average of ad Clicks = Count of ad Clicks / Count of ad Views
4. Adjusted Average (with constant) = Count of ad Clicks / (Count of ad Views + constant)
5. Adjusted Average (using power) = (Count of ad Clicks)^2 / Count of ad Views

The first solution is misleading since an ad can have a huge number of views but very few clicks. The second solution depends on the count of clicks of an ad which makes sense as the likelihood of an ad to be clicked can be approximately determined from the number of times it was clicked. The third solution makes sense too, but proves wrong in case the ad click count and ad view count are both equal and low in value (eg. 2) making the probability of the ad being clicked to be 1, whereas another ad has click count of 800 and view count of 1000 making the probability of it being clicked to be 0.8. This probability is clearly misleading since the ad with higher click count should be preferred. The fourth and fifth solutions are nearly the same, both resolving the flaw of the third solution. The fourth solution penalizes ads with a small number of clicks by adding a fixed constant (usually the average of number of views). This was our first submission on Kaggle that fetched us a score if *0.62959*. The constant can be tuned to improve the score. The fifth solution focuses at rewarding the ads with large number of clicks by powering the count of clicks. This was our second submission that unfortunately fetched us a relatively lower score of *0.58414*. We realized rewarding ads by powering it's click counts provided inaccurate results, the solution was debarred.

As mentioned above our primary approach to recommend ads solely depends on the past *adclick* frequency. We then planned to incorporate more features for establishing relation between ads. Analyzing the *promoted_content.csv* file, we noticed the ads were directly mapped to *campaign_id* & *advertiser_id* in a many-to-one relation (since the same *campaign_id* or *advertiser_id* can have multiple *ad_id's*). To use them as nominal variables we created new features *campaign_id_score* and *advertiser_id_score*. These scores were simply depended on the click frequency of ads associated with them in the training set. Model based on these

Sr. No.	Features				Model		Kaggle Score
	Clicked Probability score	Advertiser score	Campaigner score	Platform score	Naïve Bayes Classifier	SGD Classifier	
1	✓	-	-	-	-	-	0.62959
2	✓	-	-	-	-	-	0.58414
3	✓	✓	✓	-	75.69%	80.03%	0.63373
4	-	-	-	✓	76.56%	80.34%	0.63798
5	-	✓	✓	✓	77.72%	80.98%	0.63861

Figure 14. Progress Report

features lead us to improved accuracy and kaggle score of *0.63373*.

We further analysed the *Clicks_train.csv* & *events.csv*, and noticed that the *display_id* in both the files had a one-to-one relation. Thus, features like *uuid*, *timestamp*, *platform* and *geo − location* could be used to further enhance our model. On analysing the feature *uuid*, we found out nearly 88% of the users had clicked ads just once. Furthermore, around 0.01112% users clicked the same ad more than once. These stats suggested using this feature would not affect the accuracy of our model extensively. Analysing the *timestamp* feature, proved it to be too complex for implementation. Next, our analysis of the feature *platform*, showed just three unique values *(1: Desktop, 2: Mobile, 3: Tablet)*. Therefore, sorting on just 3 levels was feasible. We included this feature, and our submission fetched us a score of *0.63861*. We next planned to add the feature *geo − location*, but it's analysis proved that it had a lot of missing data. Predicting how to fill in the missing data was too complex.

Next, we planned to implement the features *topic_id*, *category_id*, and *entity_id*. Each unique *document_id* were mapped to multiple *topic_id's*, *category_id's*, and *entity_id's*. We had to map a *topic_id* from *document_entities.csv* to an *ad_id* from *Clicks_train.csv*. We chose the *topic_id* with maximum *confidencelevel*, which was mapped to a *document_id* mapped it to a unique *diplay_id* from *events.csv*, which was then mapped to an *ad_id* through *Clicks_train.csv*. We followed the same methodology for *category_id* & *entity_id*. Then we calculated a score based on whether the *ad_id* mapped to the topic, entity or category, was clicked. We tried creating the master dataset including all the features like topic score, category score and entity score and the earlier ones. We couldn't merge the entire dataset with all features due to incapability of system to handle such huge data.

Next, we planned to include the features, *publisher_id* and *source_id*. Due to lot of missing values,we did not incorporated them as they would have resulted in inaccurate predictions due to mean values for most of the data. So, our final prediction model for now would predict ad click probability based on transformed features which were derived from platform, advertiser_id & campaign_id.

Currently for the above designed model we have secured a rank of 154 (top 20%) in Kaggle competition.

3.1 Build Specifications

- Processor: Intel(R) Core(TM) i7-6500U CPU @ 2.50GHz
- Memory: 16.0 GB
- Software Platform: Python 3.5
- OS: Windows 10 (64-bit)

4. Conclusion

This report presented an approach to mining logfiles of Outbrain with the goal of improving their prediction logic behind the pieces of content (ads) likely to be clicked by global users automatically. The key insight is that such clickthrough data if used as training data can lead to build efficient prediction algorithms. Based on a new formulation of the learning problem in content ranking, this report uses classification models for learning a ranking function. Taking a Naïve Bayes and SGD approach, the resulting training problem is tractable even for large numbers of events and large numbers of features.

5. Future Work

To enhance the performance of the model we can include the features *entity_id*, *category_id* and *topic_id* by merging the data with PySpark or Google BigQuery. In case the missing values of publisher id & source id can be handled, the model would predict ad clicks better. We analysed the geo-location data to deduce 80% of the ad-click data was from US. The predictions can be further narrowed down to locations based on states. Timestamp & traffic_source can also be used further enhance our prediction model.

We can implement one of the following 'probability matching' approaches/algorithms for click predictions: Upper Confidence bound FTRL-Proximal online learning algorithm

Implement and analyze the results for various modelling techniques that are used to learn a response predictor from clicks or conversions logs such as Field-aware Factorization Machines – which uses SSE instructions to accelerate vector operations and cross validation for parameter selection.

The discussed models incorporating more features can be executed in batch-process or on SQL server with large capacity to handle the data merging issue, so that feature set can be utilize to the optimum.

Clearly, there is a trade-off between the amount of training data(large no of events) and maximum homogeneity (ie. properties of a document). Is it possible to use clustering algorithms to find homogenous groups of advertisements and implement ranking function?

Furthermore, can clickthrough data also be used to adapt a search engine not to statistics of past ads selection only but to the properties of a particular document collection?

It might also be possible to explore mechanisms that make the algorithm robust against "spamming". Currently due to data handling issue for the Page Views data, it's not clear how far a single user could maliciously influence the ranking function by repeatedly clicking on particular ads.

Regarding algorithms for solving the optimization problem (currently time-consuming due to lack of optimization in data handling), it seems likely that they can be further speed up using online algorithms which are most appropriate in many application settings.

Acknowledgments

Many thanks to Prof. Mehmet Dalkilic at Indiana University Bloomington for his academic as well as professional guidance. We would also like thank Hasan Kurban and Marcin Malec for guiding us along our project milestones.

References

[1] Agichtein, E., Brill, E., Dumais, S., and Ragno, R. Learning User Interaction Models for Predicting Web Search Result Preferences. In SIGIR 2006.

[2] Joachims, T. Optimizing search engines using clickthrough data. In SIGKDD 2002

[3] Richardson, M., Dominowska, E., and Ragno, R. Predicting clicks: estimating the click-through rate for new ads. In WWW 2007

[4] Goel, A. and Munagala, K. Hybrid Keyword Search Auctions. In WWW 2009.)

[5] Richardson, M., Dominowska, E., & Ragno, R. (2007). Predicting Clicks: Estimating the Click-Through Rate for New Ads. Alberta: IW3C2.

[6] Cheng, H., & Cantu-Paz, E. (2010). Personalized CLick Prediction in Sponsored Search. New York City.

[7] Kim, S., Qin, T., Liu, T.-Y., & Yu, H. (2011). Advertiser-Centric Approach to Understand User Click Behavior in Sponsored Search. Glasgow: CIKM'11.

[8] Stochastic Gradient Descent. scikit learn. http://scikit-learn.org/stable/modules/sgd.html#stochastic-gradient-descent-for-sparse-data

[9] Outbrain Click Prediction Data https://www.kaggle.com/c/outbrain-click-prediction/data