Counterfactual Estimation and Optimization of Click Metrics in Search Engines: A Case Study

Lihong Li[1] Shunbao Chen[1] Jim Kleban[2]* Ankur Gupta[1]
[1]Microsoft Corp.
Redmond, WA 98052
[2]Facebook Inc.
Seattle, WA 98101

ABSTRACT

Optimizing an interactive system against a predefined online metric is particularly challenging, especially when the metric is computed from user feedback such as clicks and payments. The key challenge is the *counterfactual* nature: in the case of Web search, any change to a component of the search engine may result in a different search result page for the same query, but we normally cannot infer reliably from search log how users would react to the new result page. Consequently, it appears impossible to accurately estimate online metrics that depend on user feedback, unless the new engine is actually run to serve live users and compared with a baseline in a controlled experiment. This approach, while valid and successful, is unfortunately expensive and time-consuming. In this paper, we propose to address this problem using causal inference techniques, under the contextual-bandit framework. This approach effectively allows one to run potentially many *online* experiments *offline* from search log, making it possible to estimate and optimize online metrics quickly and inexpensively. Focusing on an important component in a commercial search engine, we show how these ideas can be instantiated and applied, and obtain very promising results that suggest the wide applicability of these techniques.

Categories and Subject Descriptors

G.3 [**Mathematics of Computing**]: Probability and Statistics—*Experimental Design*; H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval; H.3.3 [**Information Storage and Retrieval**]: Online Information Services

General Terms

Experimentation, Performance

Keywords

Experimental design, counterfactual analysis, Web search, query correction/rewriting, information retrieval, contextual bandits

*This work was done when J. Kleban was with Microsoft.

WWW '15 Florence, Italy
ACM 978-1-4503-3473-0/15/05.
http://dx.doi.org/10.1145/2740908.2742562.

1. INTRODUCTION

The standard approach to evaluating ranking quality of a search engine is to evaluate its ranking results on a set of human-labeled examples and compute *relevance metrics* like mean average precision [1] and normalized discounted cumulative gain (NDCG) [15]. Such an approach has been highly successful at facilitating easy comparison and improvement of ranking functions (*e.g.*, [32]).

However, such offline relevance metrics have a few limitations. First, there can be a mismatch between users' actual information need and the relevance judgments. For example, for the query "tom cruise," it is natural for a judge to give a high relevance score to the actor's official website, http://tomcruise.com. However, search log from a commercial search engine may suggest the opposite—users who issue that query are often more interested in news about the actor, not the official website. Second, in some applications like personalized search [24], judges often lack necessary information to provide sensible labels. Third, user experience with a search engine can depend on other modules like user interfaces. Relevance labels for query-document pairs only reflect one aspect of a search engine's overall quality. Finally, an important factor in nowadays search engine is its monetization performance (from advertising), which cannot be easily judged by human labelers.

All the challenges above imply the strong need for considering user feedback in evaluating, and potentially optimizing, a search engine. For example, user behavior like clicks is used to infer personalized relevance for evaluation purposes [31], and to compare two ranking systems by interleaving [6].

Unfortunately, metrics that depend on user feedback are hard to estimate *offline*, due to their counterfactual nature. For example, suppose we are interested in measuring the time-to-first-click metric. When we change any part of the search engine, the final search engine result page (SERP) for a particular query may be different, and hence users' click behavior may change as well. Based on search log, it is often challenging to infer what a user would have done for a SERP different from the one in the log. Prediction errors of state-of-the-art user click models (*e.g.*, [7, 12] and their variants) are likely much larger than usual improvements of click-based metrics in nowadays commercial search engines that are already highly optimized. Therefore, offline evaluation based on such user models may not always be reliable.

In practice, the common solution is to run a controlled experiment (*a.k.a.* an A/B test). Specifically, one randomly splits users into two statistically identical groups, known as *control* and *treatment*, respectively. Users in the control group are served by a baseline search engine, while users in the treatment group by a modified engine (which often differs from the baseline in one component). The experiment usually lasts for days or weeks, at the end of which online metrics (like click-through rate and time to first click) of the

two systems are calculated. One then applies statistical hypothesis testing to conclude whether the modified engine is better than the baseline at a certain significance level.

Controlled experiments have proved very successful in practice (*e.g.*, [16]), allowing engineering and business decisions to be made in a data-driven manner. However, these experiments usually require nontrivial engineering resources and are time-consuming, since the experiments are run on *real* users, so significant efforts are needed to avoid surprises in the experiments. Furthermore, when trying to optimize a click metric, one often takes a guess-then-check approach: an easy-to-compute *proxy* metric (like NDCG) is used offline to obtain a modified (and hopefully better) engine, whose online metric improvement (if any) is verified in an A/B test. Due to its approximation nature, the offline proxy metric can be misleading in determining which modified system to run in the experiment. Combined with the long turnaround time of A/B tests, such an *indirect* optimization procedure can be rather inefficient.

In this paper, we advocate the use of causal inference techniques from statistics to perform *unbiased offline evaluation of click metrics* for search engines. Compared to A/B tests, offline evaluation allows multiple models to be evaluated on the *same* search log, without the need to be run online. Effectively, this technique makes it possible to run many A/B tests simultaneously, leading to substantial increase in experimentation agility, and to even optimize against the online metrics directly. First, we describe in Section 2 the contextual bandit as a general framework to capture many interactive problemss encountered in Web search. Section 3 presents the basic technical idea of offline evaluation, and discusses solutions to a few important practical issues. Section 4 gives details in a case study in a commercial search engine. Section 5 discusses related work. Finally, Section 6 concludes the paper. Interested readers are referred to a longer technical report [19] for more details.

2. FORMULATION

The contextual-bandit formalism [18] has proved useful in many important applications with interactions, such as online advertising [18] and content recommendation [20]. Formally, we define by $\mathcal{A} = \{1, 2, \ldots, K\}$ a set of *actions*, and by $\mathcal{X}$ a set of contextual information. A contextual bandit describes a round-by-round interaction between a learner and the environment: at each round,

- Environment chooses contextual information x, and a *reward* signal r_a for each $a \in \mathcal{A}$. Only x is revealed to the learner.
- Upon observing x, the learner chooses an action $a \in \mathcal{A}$, and observes the corresponding reward r_a (but not other $r_{a'}$).

Here, we assume $r_a \in [0, 1]$ and that $(x, \vec{r})$ is drawn i.i.d. from some unknown distribution D, where $\vec{r} = (r_1, \ldots, r_K)$ is the reward vector. A common goal for a contextual-bandit learning algorithm is to optimize its action-selection policy, denoted $\pi : \mathcal{X} \to \mathcal{A}$, in order to maximize the expected reward it gets through interaction with the environment. For convenience, we call

$$V(\pi) := \mathbf{E}_{(x,\vec{r})\sim D}\left[r_{\pi(x)}\right]$$

the *value* of policy π, which measures how much per-round reward it receives on average. If we run π to choose actions for T rounds and observe the corresponding reward in every round, the value of π can be estimated simply by averaging the observed reward, and this estimate will converge to $V(\pi)$ as T increases.

As an example, consider federated search where a search engine needs to decide, given a query, whether (and where) to include vertical search results like news and images on the final SERP (*e.g.*, [9]). Here, the context x contains the submitted query, user profile, and possibly other information. The actions are how to combine the vertical search result with Web search results. The reward is often click-through (or its variants) of the vertical results. Finally, a basic problem in federate search is to optimize the policy that decides what to do with vertical search results given current context in order to maximize reward. In Section 4, we will study in more details another important component of a search engine.

An important observation in contextual bandits is that, only rewards of chosen actions are observed. Therefore, data in a contextual bandit is often in the form of (x, a, r_a). If this data is used to evaluate a policy π, which chooses a *different* action $\pi(x) \neq a$, then we simply do not have the reward signal to evaluate the policy in that context. This paper is to study how unbiasded offline policy evaluation can be done in modern search engines.

3. UNBIASED OFFLINE EVALUATION

3.1 Basic Techniques

As is observed previously [4], offline policy evaluation can be interpreted as a causal inference problem, an important research topic in statistics. Here, we try to infer the average reward $V(\pi)$ (the causal effect) if policy π is used to choose actions in the bandit problem (the intervention). The approach we take in this paper relies on randomized data collection, which has been known to be a critical condition for drawing *valid* causal conclusions. Randomzied data collection proceeds as follows: at each round,

- Environment chooses $(x, \vec{r})$ i.i.d. from some unknown distribution D, and only reveals context x.
- Based on x, one computes (*c.f.*, Section 3.2) a multinomial distribution $\vec{p} := (p_1, p_2, \ldots, p_K)$ over the actions $\mathcal{A}$. A random action a is drawn according to the distribution, and the corresponding reward r_a and probability p_a are logged.

At the end of the process, we have a set $\mathcal{D}$, called "exploration data," containing data of the form (x, a, r_a, p_a). In statistics, the probabilities p_a are also known as *propensity scores*.

When we are to evaluate the value of an arbitrary policy π *offline* using the exploration data $\mathcal{D}$, the following estimator may be used:

$$\hat{V}_{\text{offline}}(\pi) := \sum_{(x,a,r_a,p_a)\in\mathcal{D}} \frac{r_a \cdot \mathbb{I}\left(\pi(x) = a\right)}{p_a}, \tag{1}$$

where $\mathbb{I}(C)$ is the set-indicator function that evaluates to 1 if condition C holds true, and 0 otherwise.

The key benefit of this estimator is that it is unbiased (see, *e.g.*, [4, 29]): $\mathbf{E}_{\mathcal{D}}\left[\hat{V}_{\text{offline}}(\pi)\right] = V(\pi)$ for any π, provided that every component in $\vec{p}$ is nonzero. In other words, as long as we can randomize action selection, we can construct an unbiased estimate of any policy's value without even running it online on live users. Such a property holds even for certain non-static metrics, such as those that fluctuate periodically. This benefit is highly desirable, since the offline evaluator allows one to simulate many A/B tests in a fast, inexpensive, and *reliable* way.

3.2 How to Randomize Data

The unbiasedness guarantee holds for any probability distribution $\vec{p}$, as long as none of its components is zero. However, *variance* of our offline evaluator depends critically on this distribution. The evaluator gives more accurate (and thus more reliable) estimates when variance is lower.

Calculations show that the offline evaluator has a variance

$$\mathbf{Var}\left[\hat{V}_{\text{offline}}(\pi)\right] = \mathbf{E}_{(x,\vec{r})\sim D}\left[r^2_{\pi(x)}\left(\frac{1}{p_{\pi(x)}} - 1\right)\right].$$

Therefore, the variance is smaller when we place more probability mass to actions that are chosen by policy π. In reality, however,

one typically does not know π ahead of time when data are being collected, and there may be multiple policies to be evaluated on the same exploration data. One natural choice, as adopted by some authors [20, 21], is to minimize the *worst-case* variance, leading to a uniform distribution: $p_a \equiv 1/K$ for all a.

There are at least two limitations for this choice. First, choosing an action uniformly at random may be too risky for user experience, unless one knows *a priori* that every action is reasonably good. Second, when improving an existing policy that is already working reasonably well, it is likely that any improvement does not differ too much from it. Minimizing the worst-case variance may not yield the best variance reduction in reality. These two concerns imply a more conservative data collection procedure that can be more effective than the uniform random distribution. Intuitively, given a baseline policy (like the existing policy in production), we may inject randomization to it to generate randomized actions that are close to the baseline policy. The precise implementation of this idea depends on the problem at hand. In Section 4.2, we describe a sensible approach that has worked well, which is expected to be useful in other scenarios.

Finally, in order to prevent unbounded variance, it is helpful to enforce a lower bound $p_{\min}$ when computing $\vec{p}$. If such a condition cannot be met (say, due to system constraints), one can still use $\max\{p_{\min}, p_a\}$ to replace p_a in Equation 1, as done by other authors [29]. Such a thresholding trick may introduce a small bias to the offline estimator, but can drastically decrease its variance so that the overall mean squared error is reduced.

3.3 How to Verify Propensity Scores

Use of Equation 1 requires logging the probabilities p_a. Any error in calculating and logging them can lead to a bias in the final offline estimator. Furthermore, since the *reciprocal* of the score is used in the estimate, even a small error in the score can lead to a much larger bias in the offline estimator when p_a is close to 0. In a complex system, it is often nontrivial to get the scores correct. It is thus important to verify propensity scores before trusting the offline estimates. In our experience, this verification turns out to be one of the most critical, and sometimes challenging, steps when doing offline evaluation.

One solution[1] is to obtain and log a randomization seed whenever an action is chosen. Specifically, in each round of data collection (*e.g.*, Section 3.2), we choose a seed s (which may be a hashing function of context x and timestamp, *etc.*) and use it to reset the internal state of a pseudo-random number generator. Then, we use the generator to select a random action from the multinomial distribution $\vec{p}$. The final data have the form of (x, s, a, p_a, r_a), containing one more component, s. When verifying data offline, we may simply use the seed s to repeat the randomized data collection process, and check consistency among x, s, a and p_a.

An alternative, somewhat simpler approach does not require resetting the pseudo-random number generator in every round of data collection. Instead, it runs simple statistical tests to detect inconsistency. Such an approach has been quite useful in our experience, although we note that it only detects some but not all data issues.

One such test, which we call an *arithmetic mean test*, is to compare the number of times a particular action $a^* \in \mathcal{A}$ appears in the data to the expected number of occurrences conditioned on the logged propensity scores. Standard hypothesis tests can be used to estimate whether the gap between the two quantities is statistically significant or not. If the gap is significant, it indicates errors in the randomized data collection process.

[1]Proposed by Leon Bottou *et al.*, in private communication.

Another test, which we found useful, is based on the following observation: for any context x and action a^*, $\mathbf{E}_{a\sim\vec{p}}\left[\frac{\mathbb{I}(a=a^*)}{p_{a^*}} + \frac{\mathbb{I}(a\neq a^*)}{1-p_{a^*}}\right] \equiv 2$. Therefore, we may compare the mean of the above random variable from the data, and verify if it is close to the expected value, 2, using statistical hypothesis tests. Since the condition above uses harmonic means of propensity scores, we call it a *harmonic mean test*.

3.4 How to Construct Confidence Intervals

Equation 1 gives a *point*-estimate, which in itself is not very useful without considering variance information: when we compare the offline estimates of two policies' values, we must resort to reliable confidence intervals to infer whether the difference in the two point-estimates are significant or not.

Based on various concentration inequalities, Bottou *et al.* [4] developed a series of interesting confidence intervals that give useful insights. In practice, however, as suggested by the same authors, it may be better to use normal approximation to obtain confidence intervals, which is the approach we take here. Specifically, from the exploration data set $\mathcal{D}$, we can compute the standard, unbiased estimate of the standard deviation $\hat{\sigma}$ of the random variable $\frac{r_a\mathbb{I}(\pi(x)=a)}{p_a}$ from data $\mathcal{D}$. Then, a 95% confidence interval can be constructed: $\hat{V}_{\text{offline}} \pm \frac{1.96\times\hat{\sigma}}{\sqrt{|\mathcal{D}|}}$, for instance. Such an approximation works well when data size $|\mathcal{D}|$ is large and the ratios r_a/p_a are bounded.

4. CASE STUDY

4.1 Speller

Speller is a critical component in a search engine, enabling it to translate queries with typing and phonetic errors to their correct forms, so that it can match and rank relevant Web results and instant answers even when user-typed query is misspelt. Spelling correction for web queries is a hard problem, particularly because of absence of a dictionary of terms and new words and entities emerging on the Web as you are reading this. Further, one person's typo could be another person's correct query. For example, given that a user typed "CCN" has both the possibility of him wanting to type "CNN" but ending up making a typo to "CCN," or really having an intent to type "CCN." Typically, noisy channel models are applied to address this by computing probabilities of each of them being the true intents, given that "CCN" was typed, using popularities of "CCN" and "CNN" as well as how likely a user is to make this exact typo. With additional features, machine learning can be used to rank these candidates [8, 11].

The problem we focus on here is to train a model to select a subset of candidates off an already computed set of rewritten queries for a given input query. The idea to select multiple such candidates is to mitigate the risk of picking a bad correction early in the lifetime of the query [28]. After fetching the results for each of these formulations, we would either predict a single best rewrite or merge the results of multiple such rewrites.

While training the ranker of rewritten query candidates via human-labeled corrections works for a large number of queries, in cases such as above, a judge may be at a complete loss on what the users' real intent was or predicting what is the likelihood of the two intents CCN and CNN. Hence, it is desirable to learn which spelling correction actually serves the user's intent implicitly from user behavior. Furthermore, this approach is much cheaper than a human judging the queries offline. For a given spelling correction algorithm, the user's satisfaction can be measured by modeling how the user interacted with the search engine on a real search session.

Having said that, every single technique or algorithm in the improvement of a search algorithm cannot be exposed to the user to measure its goodness, given that: (1) It can pose risk to the relevance and quality of results seen by the users in the experiment. (2) User query volume restricts total number of experiments that can be run per unit time. (3) Cost of failure is higher online since typically more investment is required to make code robust enough to bring it in front of the users. (4) Online experimentation results are noisier and harder to analyze against baseline, since the queries, users and sessions on which two algorithms were run differ. These concerns lead to a need of an offline evaluation system even when the labels at first place have been collected via online data.

Using terminology in Section 2, the context includes the user-typed query and rewritten candidates (together with their features); an action is a *subset* of candidate rewrites; and the reward is a metric derived from user clicks on the final search result page. Due to business sensitivity, the metric is not revealed, although it suffices to say that the metric measures goodness of the final SERP for the present user. From now on, this metric is referred to as the *target metric*. Apparently, the actions affect the final SERP, which in turn affect user clicks and the reward.

4.2 Data Collection

Our data collection process was run on a small fraction of random users (identified by browser cookies) for a week in late 2013, yielding about 15M impressions. This is the exploration data set $\mathcal{D}$ to be used in the following experiments.

To avoid adverse user experience, we require the top candidate must be included in the action for every impression. Other candidates were sent randomly: for $i \in \{2, \ldots, L\}$, $\Pr(q_i \text{ is chosen}) := \frac{1}{1+\exp(\lambda_1(s_1 - s_i)+\lambda_2)}$, where s_i was a score indicating how good candidate q_i was as a reformulation of the original query; λ_1 and λ_2 were parameters tuned to yield a good balance between aggressiveness of exploration and potential negative user experience. In our case, the parameters were chosen so that the selection probabilities $\Pr(q_i \text{ is chosen})$ fell in $[0.1, 0.9]$, and that an offline metric based on judge labels was not severely affected. With these probabilities, propensity scores of actions (each being a subset of query candidates) can be computed easily.

There are a few benefits of using this randomization scheme. First, by always including q_1 in P, the final SERP is usually reasonably good, so users who were included in the data-collection process would not notice much decrease in relevance quality. Second, the scheme is motivated by the intuition that candidates with higher scores tend to be better, so are more likely to be chosen by a good policy. Biasing data collection towards such more promising candidates will likely reduce variance (*c.f.*, Section 3.2).

After data were collected, we ran the arithmetic and harmonic mean tests described in Section 3.3 and found no major issues. It is worth mentioning that the tests were able to help detect data issues in earlier versions of data collection, leading to fixes that eventually improved data quality.

4.3 Bootstrapping Histogram

In Section 3.4, we advocate the use of asymptotic normal approximation to construct confidence intervals. The underlying assumption is that, when the amount of data is large, the estimator in Equation 1 is almost normally distributed. Here, we verify this assumption empirically with bootstrapping.

In particular, we sampled impressions from $\mathcal{D}$ with replacement to get a bootstrapping set $\mathcal{D}_1$ of the same size as $\mathcal{D}$. The sampling procedure was repeated $B = 1000$ times, and we ended up with B data sets: $\mathcal{D}_1, \mathcal{D}_2, \ldots, \mathcal{D}_B$. On each of them, we computed the unbiased offline estimate of click-through rate (CTR) for some fixed policy π. Finally, we had B estimates of CTR for the same policy. Since we were dealing with a large data set, we implemented the *online* bootstrap [23], which is essentially identical to the standard bootstrap for the size of our data.

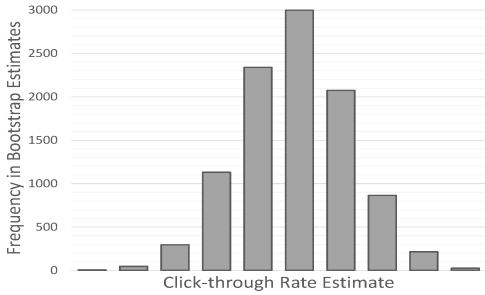

Figure 1: Bootstrapping histogram. The x-axis is the offline CTR estimate. The y-axis is the frequency of the bootstrapped estimates that fall into the respective range of CTR.

Figure 1 gives the histogram of the B estimates of CTR.[2] It is rather close to a normal distribution, as expected. The result thus validates the form of confidence intervals given in Section 3.4. Given the size of our data, the confidence intervals computed this way were all tiny, usually on the order of 10^{-2} or less. We therefore did not show these tiny intervals in the plots in the paper.

4.4 Accuracy against Ground Truths

We now investigate accuracy of the offline evaluator. During the same period of time when $\mathcal{D}$ was collected, we also ran a candidate selection policy π on a fraction of users. The online statistics of π could then be used as "ground truth," which can be used to validate accuracy of the offline estimator using exploration data $\mathcal{D}$.

First, we examine the estimate of the target metric for each day of the week. Figure 2 is a scatter plot of the online (true) vs. offline (estimated) values. As expected, the offline estimates are highly accurate, centering around the online ground truth values. Also included in the plot is a biased version of the offline estimate, labeled "Offline (biased)," which uses the following variant of Equation 1:

$$\hat{V}^{(\text{biased})}_{\text{offline}}(\pi) := \frac{\sum_{(x,a,r_a,p_a)\in\mathcal{D}} r_a \mathbb{I}(\pi(x) = a)}{\sum_{(x,a,r_a,p_a)\in\mathcal{D}} \mathbb{I}(\pi(x) = a)}.$$

This estimator, which is not uncommon in practice, ignores sampling bias in $\mathcal{D}$. The effect can be seen from the much larger estimation error compared to the online ground truths, confirming the need for using the reciprocal of propensity scores in Equation 1.

We now look at the CTR metric more closely. Figure 3 plots how daily CTR varies within one week. Again, the offline estimates match the online values very accurately. The gap between the two curves is not statistically significant, as the 95% confidence intervals' widths are roughly 0.01. Figure 4 gives the fraction of clicks contributed by URLs of different positions on the SERP. Figure 5 measures how many clicks were received within a given time after a user submitted the query. All results show high accuracy of the offline evaluator. We have also done the same comparison on other metrics and observed similar results.

4.5 Offline Optimization

[2]All CTRs and target metrics reported in the paper are normalized (*i.e.*, multiplied by a constant) for confidentiality reasons.

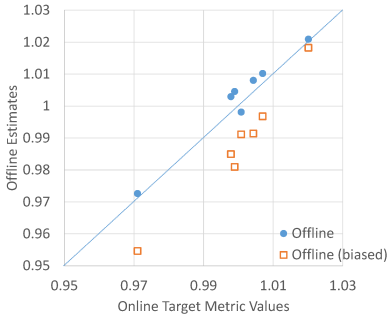

Figure 2: Online vs. offline target metric values. Each point corresponds to one of the 7 days in the data collection period.

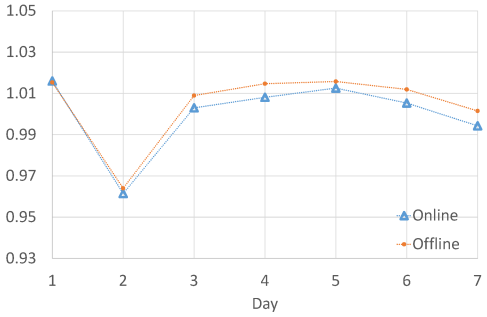

Figure 3: Daily click-through rates (CTR).

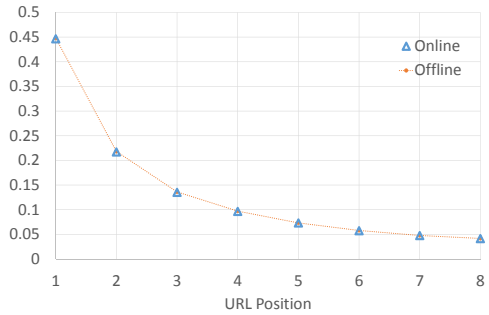

Figure 4: CTR as a function of position on SERP.

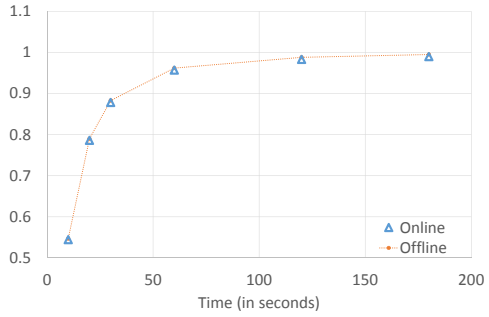

Figure 5: CTR as a function of time (in seconds).

While offline evaluation can be very useful on its own, reducing a substantial fraction of online A/B testing, it can be used as a subroutine for offline *optimization* of policies.

We used a different data set similarly collected from fall 2013, and split it randomly into a training set $(2/3)$ and an evaluation set $(1/3)$. From the training set, we extracted training data whose (binary) labels were determined based on whether the sent candidate contributed positively to the target metric. Hence, we obtained a binary classification problem, in which one tries to predict whether a rewritten query will contribute to the target metric, conditioned on the original query. Boosted logistic regression was used to learn a model, which outputs a number for every query-candidate pair.

When learning such a model, a few meta-parameters had to be chosen to balance the target metric improvement and capacity constraints (since sending too many rewrites can be expensive). Here, we did a grid search on the most important parameter. For each value, we used the unbiased offline evaluation (Equation 1) to predict its online target metric, and selected the best one that respects capacity constraints. Based on the offline evaluation results, we picked a model and ran an A/B test. In a two-week experiment done in late 2013, the model did show a statistically significant improvement over the existing (already highly optimized) production baseline, demonstrating the power of the offline evaluator. Interested readers are referred to our technical report [19] for more details of the optimization step and a few successful cases where the new model improves upon the baseline.

5. RELATED WORK

There is a long history of evaluation methodology research in the information retrieval community [27]. The dominant approach is to collect relevance judgments for a collection of query-document pairs, and compare different retrieval/ranking functions against metrics like mean average precision and normalized discounted cumulative gains. This approach has been very successful as a low-cost evaluation scheme. However, several authors have argued for several of its limitations (*e.g.*, [2, 33]), in addition to the ones discussed in Sections 1 and 4.1; an alternative, "user-centered" evaluation emphasizes interaction between user and the search engine. One challenge with the user-centered approach is the relatively high cost for system evaluation and comparison. Our work, therefore, demonstrates a promising solution in a large-scale search engine.

In industry, people also measure various online metrics (*e.g.* CTR) to monitor and compare systems while running them to serve users. Randomized control experiments (*e.g.*, [16]) are the standard way to measure and compare such online metrics. The more recent interleaving technique may also be used to quickly identify the winner between competing systems [6]. Both techniques requires running experiments on users, while our offline approach here can be more efficient and less expensive. An interesting exception is the recent work [13] that applies unbiased offline evaluation to infer interleaving outcomes. It should be noted that, for evaluating a single policy, our approach is less efficient; however, since the same exploration data is used to evaluate *multiple* policies, it is expected to be much more data-efficient in the long run.

As mentioned earlier, the offline evaluation technique is closely related to causal inference in statistics [14] (as well as the covariate shift problem in machine learning [26]), in which one aims to infer, from observational data, the counterfactual effect on some measurement by changing the policy (more often called "intervention" in the statistics literature). Such counterfactual methods have shown promise in a few important Web applications recently [4, 5, 17, 18, 21, 29, 30, 25]. In contrast to previous work that uses a general framework [4], we focus the more specialized contextual bandit model as in [18, 29, 21], which is natural and rich enough to model many interactive machine learning problems. Furthermore, instead of adding noise to parameters in a system to collect exploration data [4], we randomize actions directly, which is more efficient when the number of actions is small. A common algorithmic tool in these works (including ours) is importance sampling for bias

correction, which requires randomization (or natural exploration) in data. They therefore share a common limitation that the variance tends to be large when the number K of actions is large.

6. CONCLUSIONS

In this work, we formulate a class of optimization problems in search engines as a contextual bandit problem, and focus on the *offline* policy evaluation problem. Our approach uses counterfactual analytic techniques to obtain an unbiased estimate of the true policy value, without the need to run the policy on real users. Using data collected from a commercial search engine, we verified the reliability of such an evaluation, and also showed a successful application of it for offline optimization.

The promising results in this paper suggest a number of interesting directions for future research. The action set in Speller is tractably small when we only consider a short list of candidates. The set of actions in a ranking problem, defined naively, consists of all permutations of URLs. This is an exponentially large set that can cause the variance to be large. It would be interesting to see how to leverage successful ideas in related work [4, 10, 22] to address this issue. Another direction worth investigating is direct optimization of policies based on exploration data with, for instance, the offset tree algorithm among other approaches [3].

Acknowledgements

We thank Leon Bottou, Chris Burges, Nick Craswell, Susan Dumais, Jianfeng Gao, John Platt, Ryen White, and Yinzhe Yu for helpful discussions.

7. REFERENCES

[1] Ricardo Baeza-Yates and Berthier Ribeiro-Neto. *Modern Information Retrieval: The Concepts and Technology behind Search*. Addison-Wesley Professional, 2nd edition, 2011.

[2] Nicholas J. Belkin. Some(what) grand challenges for information retrieval. *SIGIR Forum*, 42(1):47–54, 2008.

[3] Alina Beygelzimer and John Langford. The offset tree for learning with partial labels. In *KDD*, 2009.

[4] Léon Bottou, Jonas Peters, Joaquin Quiñonero-Candela, Denis Xavier Charles, D. Max Chickering, Elon Portugaly, Dipankar Ray, Patrice Simard, and Ed Snelson. Counterfactual reasoning and learning systems: The example of computational advertising. *JMLR*, 14:3207–3260, 2013.

[5] David Chan, Rong Ge, Ori Gershony, Tim Hesterberg, and Diane Lambert. Evaluating online ad campaigns in a pipeline: Causal models at scale. In *KDD*, 2010.

[6] Olivier Chapelle, Thorsten Joachims, Filip Radlinski, and Yisong Yue. Large scale validation and analysis of interleaved search evaluation. *ACM TOIS*, 30(1):6, 2012.

[7] Olivier Chapelle and Ya Zhang. A dynamic Bayesian network click model for Web search ranking. In *WWW*, 2009.

[8] Silviu Cucerzan and Eric Brill. Spelling correction as an iterative process that exploits the collective knowledge of Web users. In *EMNLP*, 2004.

[9] Fernando Diaz. Integration of news content into Web results. In *WSDM*, 2009.

[10] Miroslav Dudík, John Langford, and Lihong Li. Doubly robust policy evaluation and learning. In *ICML*, 2011.

[11] Jianfeng Gao, Xiaolong Li, Daniel Micol, Chris Quirk, and Xu Sun. A large scale ranker-based system for search query spelling correction. In *COLING*, 2010.

[12] Fan Guo, Chao Liu, Anitha Kannan, Tom Minka, Michael J. Taylor, Yi-Min Wang, and Christos Faloutsos. Click chain model in Web search. In *WWW*, 2009.

[13] Katja Hofmann, Anne Schuth, Shimon Whiteson, and Maarten de Rijke. Reusing historical interaction data for faster online learning to rank for IR. In *WSDM*, 2013.

[14] Paul W. Holland. Statistics and causal inference. *Journal of the American Statistical Association*, 81(6):945–960, 1986.

[15] Kalervo Järvelin and Jaana Kekäläinen. Cumulated gain-based evaluation of IR techniques. *ACM Trans. Inf. Syst.*, 20(4):422–446, 2002.

[16] Ron Kohavi, Roger Longbotham, Dan Sommerfield, and Randal M. Henne. Controlled experiments on the web: Survey and practical guide. *Data Minining and Knowledge Discovery*, 18:140–181, 2009.

[17] Diane Lambert and Daryl Pregibon. More bang for their bucks: Assessing new features for online advertisers. *SIGKDD Explorations*, 9(2):100–107, 2007.

[18] John Langford, Alexander L. Strehl, and Jennifer Wortman. Exploration scavenging. In *ICML*, 2008.

[19] Lihong Li, Shunbao Chen, Ankur Gupta, and Jim Kleban. Counterfactual analysis of click metrics for search engine optimization. Technical Report MSR-TR-2014-32, Microsoft Research, 2014.

[20] Lihong Li, Wei Chu, John Langford, and Robert E. Schapire. A contextual-bandit approach to personalized news article recommendation. In *WWW*, 2010.

[21] Lihong Li, Wei Chu, John Langford, and Xuanhui Wang. Unbiased offline evaluation of contextual-bandit-based news article recommendation algorithms. In *WSDM*, 2011.

[22] Lihong Li, Jin Kim, and Imed Zitouni. Toward predicting the outcome of an A/B experiment for search relevance. In *WSDM*, 2015.

[23] Nikunj C. Oza and Stuart Russell. Online bagging and boosting. In *AISTATS*, 2001.

[24] James Pitokow, Hinrich Schütze, Todd Cass, Rob Cooley, Don Turnbull, Andy Edmonds, Eytan Adar, and Thomas Breuel. Personalized search. *CACM*, 45(9):50–55, 2002.

[25] Ashok Kumar Ponnuswami, Kumaresh Pattabiraman, Desmond Brand, and Tapas Kanungo. Model characterization curves for federated search using click-logs: Predicting user engagement metrics for the span of feasible operating points. In *WWW*, 2011.

[26] J. Quiñonero-Candela, M. Sugiyama, A. Schwaighofer, and N. Lawrence, editors. *Covariate Shift and Local Learning by Distribution Matching*. MIT Press, 2008.

[27] Stephen Robertson. On the history of evaluation in IR. *Journal of Information Science*, 34(4):439–456, 2008.

[28] Daniel Sheldon, Milad Shokouhi, Martin Szummer, and Nick Craswell. Lambdamerge: merging the results of query reformulations. In *WSDM*, 2011.

[29] Alexander L. Strehl, John Langford, Lihong Li, and Sham M. Kakade. Learning from logged implicit exploration data. In *NIPS 23*, 2011.

[30] Liang Tang, Romer Rosales, Ajit Singh, and Deepak Agarwal. Automatic ad format selection via contextual bandits. In *CIKM*, 2013.

[31] Jaime Teevan, Susan T. Dumais, and Eric Horvitz. Potential for personalization. *ACM Trans. Comput.-Hum. Interact.*, 17(1), 2010.

[32] TREC. The Text REtrieval Conference. http://trec.nist.gov.

[33] Andrew H. Turpin and William Hersh. Why batch and user evaluations do not give the same results. In *SIGIR*, 2001.