Model Characterization Curves for Federated Search Using Click-Logs: Predicting User Engagement Metrics for the Span of Feasible Operating Points

Ashok Kumar Ponnuswami
Microsoft Corporation
One Microsoft Way
Redmond, WA 98052
asponnus@microsoft.com

Kumaresh Pattabiraman
Microsoft Corporation
One Microsoft Way
Redmond, WA 98052
kpattab@microsoft.com

Desmond Brand
Microsoft Corporation
One Microsoft Way
Redmond, WA 98052
debrand@microsoft.com

Tapas Kanungo
Microsoft Corporation
One Microsoft Way
Redmond, WA 98052
tkanungo@microsoft.com

ABSTRACT

Modern day federated search engines aggregate heterogeneous types of results from multiple vertical search engines and compose a single search engine result page (SERP). The search engine aggregates the results and produces one ranked list, constraining the vertical results to specific slots on the SERP.

The usual way to compare two ranking algorithms is to first fix their operating points (internal thresholds), and then run an online experiment that lasts multiple weeks. Online user engagement metrics are then compared to decide which algorithm is better. However, this method does not characterize and compare the behavior over the entire span of operating points. Furthermore, this time-consuming approach is not practical if we have to conduct the experiment over numerous operating points.

In this paper we propose a method of characterizing the performance of models that allows us to predict answers to "what if" questions about online user engagement using click-logs over the entire span of feasible operating points. We audition verticals at various slots on the SERP and generate click-logs. This log is then used to create operating curves between variables of interest (for example between result quality and click-through). The operating point for the system then can be chosen to achieve a specific trade-off between the variables. We apply this methodology to predict i) the online performance of two different models, ii) the impact of changing internal quality thresholds on clickthrough, iii) the behavior of introducing a new feature, iv) which machine learning loss function will give better online engagement, v) the impact of sampling distribution of head and tail queries in the training process. The results are reported on a well-known federated search engine. We validate the predictions with online experiments.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval

WWW 2011, March 28–April 1, 2011, Hyderabad, India.
ACM 978-1-4503-0632-4/11/03.

General Terms

Algorithms, Experimentation, Human Factors, Measurement, Performance, Theory

Keywords

Online evaluation, user engagement, offline prediction, randomization, metrics

1. INTRODUCTION

Current federated web search systems, like any other system, have many internal system parameters that need to be fixed prior to releasing it to real users. Fixing the operating point — the set of system parameter values — can be a time consuming process in the standard approach: i) train an algorithm, ii) fix system parameters/thresholds (operating points), iii) conduct multi-week online experiments, iv) evaluate the results.

A *characterization curve* of a system, on the other hand, provides a performance curve of a basic component over the entire span of possible operating points. For example, a transistor characteristics curve can be used to find what the the voltage will be if the current is at a specific level, and how this curve changes with temperature. This information is then used by engineers to design new circuits and predict its presentation. In a similarly way, in federated search, a change in operating point usually results in a change in the vertical rankings and hence a change in average quality metric distributions at specific positions and overall user engagement metrics. Today estimates for engagement metric impact are measured by doing online experiments over weeks of data. In this paper we present a methodology that uses randomization of results that enables us to create operating curves for the expected performance of the algorithm for all possible operating points. Furthermore it enables us to answer many design related questions such as:

- Can we predict the online behavior of the system if we change an internal threshold without performing new online experiments?
- Can we understand the online impact of a specific feature in the model at various operating points?

- Given two different machine learning loss functions that can be used for estimating models, can we show which one gives us better online user engagement metrics in an offline way?

- In any machine learning process the engineer has to struggle with issues of query sampling and weighting of training records. Is there a systematic approach to quantifying which sampling or weighting scheme leads to better user engagement profile?

In Section 2 we summarize research relevant to our work. We provide an overview of our federated web search architecture, metrics, and experimental setup in Section 3. In Section 4 we describe our characterization methodology and provide the algorithmic details. Next we demonstrate the methodology by applying it to answer numerous questions about our federated web search algorithm, which is described in Section 6.

2. RELATED LITERATURE

Receiver Operating Characterization (ROC) curves were introduced in early signal detection literature [23] and have since been used extensively in psychology [17], pattern recognition and machine learning [11, 12], computer vision [16] and numerous other fields. Essentially the idea is to characterize the performance of a system over the entire set of feasible operating points. This characterization curve can then be used later to predict behavior or explain the behavior if the operating point changes for some reason. The trade-off curves are typically between probability of mis-detection and probability of false alarm since, which are then used to compute total probability of error or cost.

In the field of information retrieval, there are numerous papers that that present plots for trade-offs of various types. For example, Collins [3, 4] presents results that show the impact of varying query expansion parameters. Craswell *et. al* [9], characterize the impact of crawl selection methods on retrieval effectiveness. While these papers discuss trade-off and offline evaluation they are not about about online performance characterization from offline experiments. Evaluation literature in information retrieval community depends heavily on sample corpus and human judgments [24, 13], which reflect the views of the human judges and not on what will happen with respect to user engagement. This is crucial since it is very difficult for a judge to know what the user wanted in the case of tail or location-specific queries.

In machine learning literature, researchers have shown how to choose optimal operating point for specific metrics. Joachim [15] shows how to optimize for for various loss functions such as ROC area, Precision/Recall Break-Even Point (PRBEP). However, while this is an offline optimization and evaluation method, it is not representing or predicting online engagement behavior of users.

Federated web search [21] is an active field of research. Si and Callan [22] discuss approaches to incorporating vertical search engine effectiveness. Diaz *et al* [1, 6] present machine learned approaches to trigger verticals and adapting the triggering based on click feedback. We [18] introduced the notion of using clicks as labels for machine learning and present online performance results. In these papers, online results are presented at only one operating point.

On-line behavior models [5, 2, 14, 7] estimated from click-logs can be used to predict the user satisfaction in online sessions. These are helpful to estimate the performance (without collecting human judgments) of an online experiment *after* it has been performed on real online traffic. Finally, the notion of randomization of result presentation has been used to collect click-preferences for web results [19]. However, this work does not propose further use for characterizing performance.

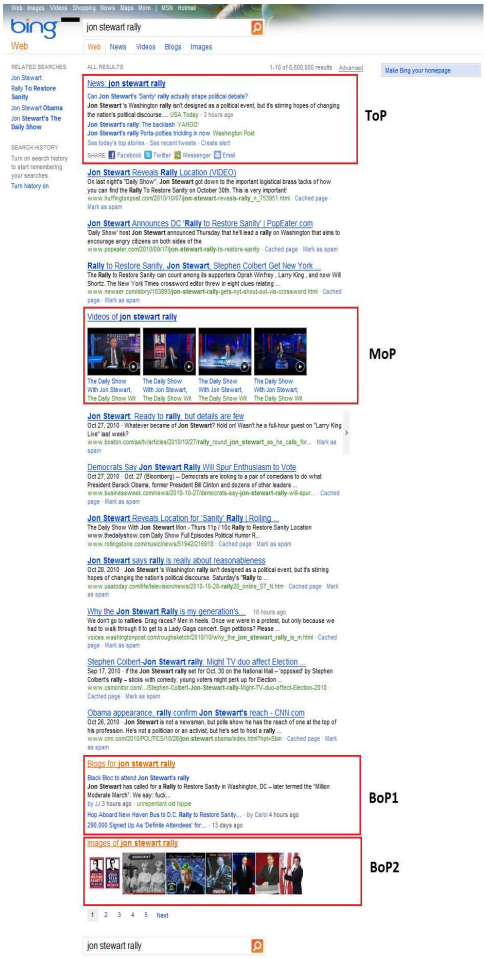

Figure 1: Screenshot of the Bing SERP showing vertical search results at different slots.

In this paper we propose a method that allows us to predict online metrics over the entire range of feasible of system operating points. This is in contrast to online experiments that i) usually take weeks to get statistically significant results, and ii) cannot be run for numerous operating point due to lack of online traffic. Collecting click-logs of online experiments where we randomize the presentation of verticals allows us to simulate online traffic for various different operating points with potentially different search result pages for the same query.

3. OVERVIEW

We start with a brief overview of how our system places verticals on the SERP and how we currently perform online experiments.

3.1 SERP Composition

When a user query arrives at the search engine, it is sent to the web result generator and to a vertical selection component that determines which verticals should be triggered. Some of the verticals that are selected at this step generate content in response to the query. The content from the verticals that respond are then sent with the web results to a ranking and placement module. This module is responsible for determining where the verticals are placed on the SERP. The SERP is then rendered and sent to the user.

Figure 1 gives an examples of a SERP rendered by Bing. Verti-

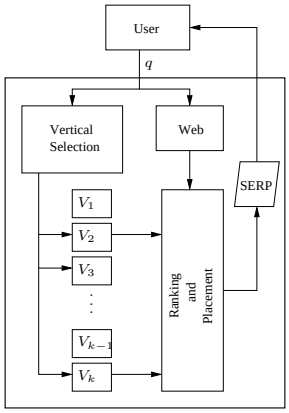

Figure 2: A high level overview of how the SERP is generated in response to a user query.

cals are frequently shown at specific slots, in part due to aesthetic appeal and often due to the way pages are rendered on the user's browser. The common slots are at the top of page (TOP), middle of page (MOP) and bottom of page (BOP). The location for the MOP slot is picked so that it appears just above the "fold" (the line on the page below which the user can not view the content without scrolling) for most users. The set of slots is denoted by $S = \{\text{TOP}, \text{MOP}, \text{BOP}\}$. The approach we describe in this paper is not constrained to any specific way of picking locations for the slots and hence we can think of the set of slots as being $S = \{s_1, s_2, \ldots, s_k\}$ (see Figure 3).

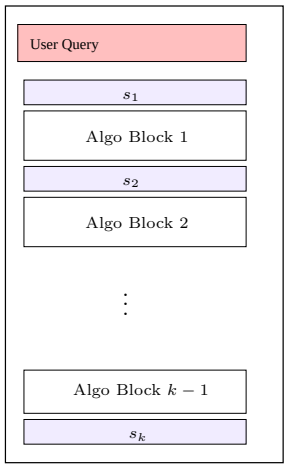

Figure 3: A simplified version of the SERP showing slots where the verticals can be placed. A fixed number of web results can be placed between two consecutive slots.

3.2 Assigning Verticals to Slots

We now mention how the ranking and placement module (Figure 2) assigns verticals to the slots. Once a subset of the verticals generate content in response to a query, they are ranked and placed as described below.

Corresponding to each vertical V there is a model χ_V that ranks it. Given the feature vector for the query q, the model outputs a score $\chi_V(q)$. The idea is that if $\chi_V(q) > \chi_V(q')$, then the vertical is more relevant for query q than q'. The vertical is assigned to the slot $\mu_V(q)$ given by

$$\mu_V(q) = \begin{cases} \text{TOP} & \text{if } \chi_V(q) \geq \chi_{V,\text{TOP}} \\ \text{MOP} & \text{if } \chi_{V,\text{TOP}} > \chi_V(q) \geq \chi_{V,\text{MOP}} \\ \text{BOP} & \text{otherwise.} \end{cases}$$

The constants $\chi_{V,\text{TOP}}$ and $\chi_{V,\text{MOP}}$ are referred to as the TOP and MOP thresholds for V respectively. In general, if the SERP has k slots, then there will be $k-1$ thresholds $\chi_{V,1}, \chi_{V,2}, \ldots, \chi_{V,k-1}$ that are needed to place the vertical on the SERP.

3.3 Generating Models to Rank Verticals

We used two approaches to build the model χ_V. The first was using human judgments and the second was using pairwise click preference judgments. The modeling technique itself was gradient boosted decision trees [10]. This work is described in more detail in our paper [18].

3.4 Flights

Whenever we need to experiment with a new ranking algorithm, we assign a small set of users to a *flight* and use this ranking algorithm to generate the SERP for this set of users. For example, if we want to evaluate a new model to rank a vertical V, we take the production ranker and replace the model used to rank V with the new one to obtain the modified ranker for the flight. We call this the *treatment* flight. The metrics for this flight are then compared to a *control* flight which uses the production ranker. We usually compare the performance of the treatment flight to the control flight over the same period of time to avoid interference from temporal changes in user behavior (for example from holidays, weekend effect, breaking-news events).

Another kind of flight that we use is an *auditioning* flight. On this flight, the users are shown the verticals that trigger for a query at a random slot on the SERP (instead of using a model score to place the vertical). This flight is mainly used to gather online user engagement behavior data for various vertical placement configurations. This allows us to predict user engagement for a specific configuration in a offline setting.

Finally, each flight is constituted of a set of impressions $\mathbb{I} = \{I_1, I_2, \ldots, I_m\}$, where each impression I contains the following pieces of information: i) the query q that generated the impression, ii) the verticals that were shown on the SERP and the slots where they were shown, iii) the components on the SERP that got clicked by the user.

3.5 Metrics

For every flight, we analyze a set of metrics for each vertical to see how they differ from the control. As mentioned before, a typical experiment involves changing the model used for ranking one vertical V. In this case, some of the metrics we can analyze are:

- *Coverage*: This is the fraction of SERP impressions for which the vertical triggered. This is typically the same on the treatment and control flight unless we also perform vertical suppression.
- *Coverage at a slot s*: This is the fraction of impressions where the vertical was shown at slot s out of all the impressions where the vertical triggered. Unlike the overall coverage of the vertical, this can be directly influenced by changing the model or thresholds used to place the vertical.
- *Clickthrough*: This is the fraction of times that the vertical was clicked out of *all* the impressions (not just those on which the vertical was shown). This is of particular interest

to our partner vertical teams since this is the amount of traffic they get. Ideally, they would like to see this metric to be as high as possible. We also frequently look at clickthrough at slot s which is the fraction of times that the vertical was clicked when it was shown at s out of all the impressions.

- *Vertical Clickthrough Rate (CTR)*: This is the fraction of times that the vertical was clicked out of the impressions where the vertical was shown on the SERP. The vertical CTR at slot s is defined as the fraction of times the vertical was clicked out of the impressions where the vertical was shown at s. Usually if we increase the vertical coverage at TOP, the vertical CTR increases, since components shown higher up on the page are more likely to be clicked. Also, if the model score positively correlates to relevance, then as we increase the coverage at TOP, the vertical CTR at TOP decreases. This is because we allow the vertical at TOP for more and more impressions where it scored lower. Thus to make comparing vertical CTRs meaningful, we must first ensure that there is a good coverage match at various slots and then compare the vertical CTRs at the slots. We then expect the flight with the better model to have higher vertical CTR at TOP and lower vertical CTR at BOP since it is more likely to assign higher scores to more relevant queries.

- *Normalized Vertical CTR*: This is the fraction of times that the vertical was clicked out of the impressions where either the vertical or some other result below it was clicked. The idea is that if the user clicks on the vertical or on some other result below it, it means that with high likelihood, the user noticed the vertical. This is based on the eye tracking study of Radlinski and Joachims [20]. The normalized vertical CTR at slot s is defined as the fraction of the number of times the vertical got a click when shown at s out of the impressions where the vertical was shown at s and either the vertical or some result below it got a click.

- *Sliding Normalized Vertical CTR at Slot s*: This is normalized CTR based on the impressions with score between $\chi_{V,s}$ and $\chi_{V,s+\delta s}$ in the limit $\delta s \longrightarrow 0$. In other words, this is the normalized CTR based on the impressions with the lowest vertical scores that were allowed to be shown at slot s. This is similar to metrics used by others [25, 2].

We formally define a metric as a function $f : \mathbb{I} \longrightarrow \mathbb{R}$. For example, the vertical clickthrough at TOP is defined as

$$\text{Clickthrough}_{V,\text{TOP}}(\mathbb{I}) = \frac{|\{I \in \mathbb{I} : \text{IsSlot}_{V,\text{TOP}}(I) \wedge \text{Click}_V(I)\}|}{|\mathbb{I}|},$$

where $\text{IsSlot}_{V,s}(I)$ is true if V was shown at slot s in I. and $\text{Click}_V(I)$ is true if the V was shown in the impression I and the user clicked on V. Define $\text{ClickBelow}(I)$ to be true if V was shown in impression I and some result below V got clicked. Then we can define normalized CTR at slot s as

$$\text{NormCTR}_{V,s}(\mathbb{I}) = \frac{|\{I \in \mathbb{I} : \text{IsSlot}_{V,s}(I) \wedge \text{Click}_V(I)\}|}{|\{I \in \mathbb{I} : \text{IsSlot}_{V,s}(I) \wedge (\text{Click}_V(I) \vee \text{ClickBelow}_V(I))\}|},$$

where χ_V is the model used to rank V and $\chi_{V,s}$ is the threshold for slot s (see Section 3.2). The sliding normalized CTR can be defined as

$$\bigtriangledown\text{NormCTR}_{V,s}(\mathbb{I}) = \lim_{\delta x \longrightarrow 0} \frac{|\{I \in \mathbb{I} : \chi_{V,s} \leq \chi_V(I) < \chi_{V,s} + \delta x \wedge \text{IsSlot}_{V,s}(I) \wedge \text{Click}_V(I)\}|}{|\{I \in \mathbb{I} : \chi_{V,s} \leq \chi_V(I) < \chi_{V,s} + \delta x \wedge \text{IsSlot}_{V,s}(I) \wedge (\text{Click}_V(I) \vee \text{ClickBelow}_V(I))\}|}.$$

A problem with the above definition of sliding normalized CTR is that we need to have infinite number of impressions, which is not realistic. In practice, we will have to approximate it by looking at a small finite window of score around the threshold. The size of this window would depend on the size of the flight and the confidence we want to have in the value we estimate.

4. MAIN APPROACH

We now describe how we estimate the various metric values offline using click logs from an auditioning flight. Suppose we have a ranker χ_V for a vertical V and we would like to estimate how the values of the online user engagement metrics change as we adjust the thresholds $\chi_{V,1}, \chi_{V,2}, \ldots, \chi_{V,k-1}$. Let $f^i_V : \mathbb{I} \longrightarrow \mathbb{R}$ $(i = 1, 2, \ldots, m)$ be a set of metrics that we are interested in estimating. Denote by $\mathbb{I}_{\text{audition}}$ the set of impressions from an auditioning flight, and $\chi_V(I)$ the score produced by the specific ranker χ_V for the impression I. For simplicity of notation define $\chi_{V,0} = \infty$ and $\chi_{V,k} = -\infty$.

For any new ranker and corresponding thresholds for each slot, we can always compute the ranker score and use the thresholds to compose a final SERP for each impression in the auditioning flight, The question is how do we predict the user engagement metrics for this specific composition. Now, since the auditioning flight has impressions that are all possible compositions of vertical placements in slots, and corresponding user engagement data, we can simulate a flight engagement data corresponding to a new ranker and thresholds by selecting only the impressions for each query in the auditioning flight that are identical to the composition generated by the new ranker and thresholds. This idea is summarized by the following proposition.

PROPOSITION 1. *Suppose on the auditioning flight, the vertical V was placed uniformly at random at one of the k slots. Also suppose that users pick queries at random from some fixed distribution and issue them to the search engine and that users interact with the SERP without any context of the previous queries they issued. Then the set $\mathbb{I} = \{I \in \mathbb{I}_{\text{audition}} : \chi_{V,i} \leq \chi_V(I) < \chi_{V,i-1}$ where I places V in slot $s_i\}$ simulates a treatment flight that ran in parallel the auditioning flight with $1/k$ fraction of the users as on the auditioning flight and where χ_V was the ranker for V and the thresholds were $\chi_{V,1}, \chi_{V,2}, \ldots, \chi_{V,k-1}$.*

PROOF. Suppose a query q was received by the search engine. Suppose $s = \mu_V(\chi_V(q))$ is the slot where we would place the vertical if we used ranker χ_V and the thresholds $\chi_{V,1}, \chi_{V,2}, \ldots, \chi_{V,k-1}$ (the treatment flight). On the auditioning flight, with probability $1/k$, it is assigned to the slot s. Therefore, every query q received on the auditioning flight has a $1/k$ probability of the vertical ending up in the same slot as on the treatment flight. Assuming then that each query is picked from the same distribution, this proves the proposition. □

Using the proposition, we can predict the metric f_V for the treatment flight to be $f_V(\mathbb{I})$, where $\mathbb{I}$ is defined in the proposition.

4.1 Predicting Metrics at TOP

Based on Proposition 1 we give the following algorithm that predicts how adjusting the TOP threshold for vertical V influences the value of various metrics.

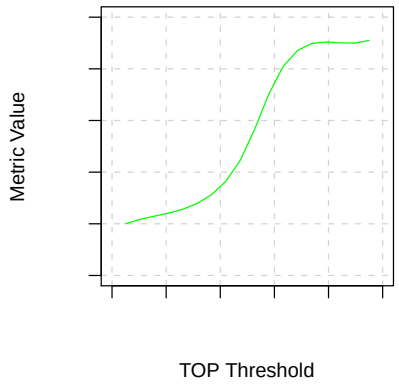

Figure 4: A plot showing a metric $f_{V,\text{TOP}}$ as a function of the TOP threshold generated using the pseudo-code in Section 4.1. This is generated by plotting $\langle \chi_V(I_i), f_{V,\text{TOP}}(\mathbb{I}_i) \rangle$.

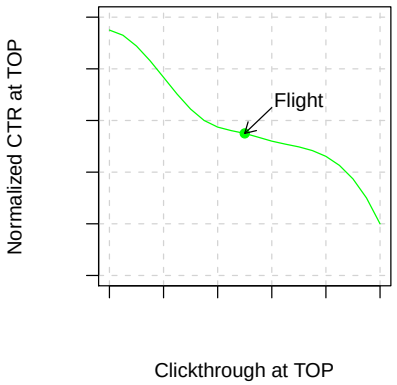

Figure 5: A plot showing the trade-off of two metrics as the TOP threshold is varied. From a flight, we only get to observe one point on this trade-off curve.

1. Retain only TOP impressions of vertical V from the auditioning flight.
2. Compute the score $\chi_V(I)$ for each impression where the vertical V triggered.
3. Sort the impressions by the score. Let $I_1, I_2, \ldots I_m$ be the sorted list of impressions.
4. For $i = 1, 2, \ldots, m$:
   (a) Let $\mathbb{I}_i = \{I_1, I_2, \ldots, I_i\}$.
   (b) Predict the value of metric $f_{V,\text{TOP}}$ to be $f_{V,\text{TOP}}(\mathbb{I}_i)$ if the TOP threshold were set to $\chi_V(I_i)$.

This gives us the user engagement metric values at TOP for the various TOP thresholds. We can now plot each metric as a function of the TOP threshold as shown in Figure 4. If we are interested in the trade-offs of two particular metrics, say $\text{Clickthrough}_{V,\text{TOP}}$ and $\text{NormCTR}_{V,\text{TOP}}$, we can also plot the trade-off curve for them. First, using the proposition, we predict $\text{Clickthrough}_{V,\text{TOP}}$ and $\text{NormCTR}_{V,\text{TOP}}$ for every possible value x of the score. With a slight abuse of notation, we represent the respective values by $\text{Clickthrough}_{V,\text{TOP}}(x)$ and $\text{NormCTR}_{V,\text{TOP}}(x)$. We can now plot the curve of $\langle \text{Clickthrough}_{V,\text{TOP}}(x), \text{NormCTR}_{V,\text{TOP}}(x) \rangle$ as a function of x as shown in Figure 5. A flight corresponds to point on this trade-off graph. If we wanted to plot the trade-off graphs using flights, we would have to flight the model with lots of different TOP thresholds and then interpolate, something that is not practical.

For the purpose of characterizing the trade-offs of a model or for comparing two different models, the metrics at TOP turn out to be the most valuable since we get a lot more click feedback at TOP (and hence the confidence in the estimates tends to be much stronger) and also since the verticals are very likely competing with the best web results.

4.2 Predicting Thresholds that Target Sliding Metrics at Various Slots

One common problem we have is controlling the quality of the verticals shown at various slots. For example, we want to show a vertical at TOP only when we expect it to be of better or comparable quality than the web results. Otherwise we want to move it down lower on the page to a slot where the value it provides is higher or comparable to the web results below it. Suppose we want to show the vertical V at the slot s for query q only if we expect the normalized CTR (or some other quality metric) at s is at or above some constant α. The problem is that a model χ_V need not necessarily predict the normalized CTR for the query at slot s. For example, a model trained on human judgments might predict a score on the same scale that the human judges used. This does not have a natural mapping to normalized CTR. Although we can train a model that predicts normalized CTR using user click logs, even in that case, we would like to see how changing α effects other metrics. Another problem of interest even if we target the metric directly might be studying the impact of changing the modeling process (for example, changing the loss function, or changing the weighing of impressions) offline without having to flight each model. We would also like to compare the trade-offs of models trained using different labels (like click logs and human judgments) and pick the best one. In order to do this, we need a function that maps the model score to the metric we are interested in.

To address these problems, we look at the "sliding" version of the quality metric we are interested in. One example was given in Section 3.5, where we defined the sliding version of the normalized CTR.

The below pseudo-code computes the $k-1$ thresholds that target a value of α for a sliding metric f:

1. For each slot $s \in S$ except the last slot on the page:
   (a) Let $\mathbb{I}_s \subseteq \mathbb{I}_{\text{audition}}$ be the set of impressions where V was in slot s from the auditioning flight.
   (b) Compute the score $\chi_V(I)$ for each $I \in \mathbb{I}_s$.
   (c) Sort the impressions by the score. Let $I_1, I_2, \ldots I_m$ be the sorted list of impressions.
   (d) For $i = w, w+1, \ldots, m$ (w is picked based on the confidence we need in the estimates):
      i. Let $\mathbb{I}_i = \{I_1, I_2, \ldots, I_i\}$. Let $\mathbb{I}'_i$ be the last w impressions from this list.
      ii. Predict that the value of f at the model score $\chi_V(I)$ is equal to $f(\mathbb{I}'_i)$ at slot s. That is, we estimate the

sliding metric f at score $\chi_V(I)$ using the sliding window of w impressions with score just greater than or equal to that score.

iii. If the estimate of f at s dips below α, output $\chi_V(I)$ as the threshold $\chi_{V,s}$ for slot s and continue to the next slot.

5. EXPERIMENTAL PROTOCOL

5.1 Auditioning

For the purposes of conducting experiments with our approach, we used an auditioning flight where 1% of the traffic was assigned to it. The impressions we used for estimating online user engagement metrics were obtained from a two week period of this flight. We present our results for three verticals, Non-Navigational (Non-Nav) News, Image and Commerce verticals.

5.2 Model Training

The models were trained on either the user click logs (click-based models) or on human judgment labels collected using our Human Relevance System (HRS) system. The models we built were using an implementation of gradient boosted decision trees [10]. We used an L2 regression loss function for the modeling, except in Section 6.5 where we study the impact of changing the loss function on online user engagement metrics. The algorithm parameter values used are described along with each experiment in the results section.

The impressions used to train our click-based models for these verticals were obtained from a two week period on the auditioning flight (this period was different from the two weeks used to gather impressions for offline user engagement metric estimation). We retained the impressions where the vertical was shown at TOP and MOP. To assign a label to each impression, the vertical was compared to the first web block (Figure 3). If the vertical was shown at TOP and got a click, but not the web block below it, the impression was labeled 1. If the vertical did not get a click when shown at TOP, but the first web block did, then the impression is labeled 0. Also, if the vertical was shown at MOP and got clicked, but not the first web block, then the impression is labeled 1, but with a higher weight (of 5). This is because the user explicitly skipped over the first web block to get to the vertical. The impressions are then log-weighed to ensure that the most frequent head queries do not completely dominate the training process. Then the impressions are re-weighed again to ensure that the head and tail both have the same weight.

The human judgment labels that we used for the News, Image and Commerce verticals had around 4,300, 39,000 and 9,000 judgments respectively. The judges were shown just the SERP with web results and the vertical content and were asked to grade the vertical on a scale of 0 to 3, indicating whether they think the vertical should not be shown in response to the query or whether the right slot is BOP, MOP or TOP respectively.

Further details of this approach are described in extensive detail in [18], but are not that relevant for the purpose of the methodology described in this paper.

5.3 Bootstrapping

Once we estimate the trade-off curves for a pair of user engagement metrics offline for two different experiments (like one that uses a click-based model and another that uses a HRS model), we would also like to know if the differences in the trade-offs are statistically significant. This is very important since we need to evaluate if we need a longer auditioning flight to make our predictions meaningful. To do so, we use bootstrapping [8]. For this, we generate n samples of the auditioning flight by sampling the impressions uniformly at random with replacement. For each sample, we calculate the value of the metrics at various scores. Suppose the offline bootstrap estimates of some metric f at a slot threshold of x for slot s are $a_1, a_2, \ldots, a_n$. Then we predict the value of the metric f at slot s with threshold x to be the median of the n values. We compute the 90% confidence interval by first sorting the a_i and then picking lower value of the interval to be at the 5th percentile of the sorted list and the higher value at the 95th percentile. Note that since in operating curves both axis are noisy, we produce both x and y confidence intervals around the estimates. We used an n of 100.

6. RESULTS AND DISCUSSION

We now demonstrate the methodology presented in Section 4 by applying it to five experiments that we conducted to answer specific questions about our federated web search algorithm.

First we explore the impact of trading-off the aggregate quality of the results that are shown to the user versus clickthrough rate. The second study is looks deeper into the impact of worst result quality (instead of aggregate) shown to the user for a specific threshold versus clickthrough rate. We next demonstrate how to use the characterization curves to study the online user engagement impact of a feature in an offline way. This is followed by an experiment where we study the impact of various query sampling strategies in the modeling process. And finally we apply the methodology to study the impact of loss functions on the online engagement metrics.

6.1 Trade-off of Clickthrough and Quality

Ideally, our vertical partners would like their verticals to be shown on as high a slot as possible. On the other hand, we want to ensure that if the web results provide more utility than the vertical, then the vertical is not shown above them. One metric of prominence that the vertical teams frequently look at is clickthrough. A metric that measures the quality of the vertical is the normalized CTR, which measures how the vertical performs compared to the web results below it. We compare how different models for some major verticals trade-off these two metrics at TOP. We use the algorithm in Section 4.1 to estimate the value of these metrics as a function of the TOP threshold and then plot the trade-off graph for each model. Using bootstrapping, we can also calculate the error margins for the estimate of both these metrics. As we decrease the TOP threshold, we let the vertical appear at TOP for more queries. This will lead to increased clickthrough to it. At the same time, we expect that since we are letting the vertical at TOP for less relevant queries (assuming the model score correlates positively to relevance), the normalized CTR will drop. This is shown in Figures 6-8 for three different models. For each curve we also show the error margins for our estimates on clickthrough and normalized CTRs at TOP using horizontal and vertical error bars. The X-axis has been scaled so that the maximum possible clickthrough to 1. The Y-axis has been linearly transformed to be between 0-1.[1]

We plot the graphs for three models for each vertical. The curve labeled "Vertical Confidence" denotes the trade-off of a model that just outputs the vertical confidence (a feature provided by the answer partner as an indicator of how relevant they think their vertical is to a query) as the score. Vertical confidence is usually one of the most important features in our click-based and HRS models, but is calculated without considering the other web results present on the SERP. But when the verticals are ranked by the ranking and placement module of the search engine, this information can be incorporated into the ranking process as features. In some sense, the

[1]This was done due to business constraints.

curves for the click-based and HRS-based models show that improved utility that this component provides. In general, one can see that for the same amount of clickthrough, the click-based model achieves a better value of normalized CTR at TOP compared to the HRS model or just using vertical confidence.

From a flight, in the past, we used to be able to see only one point $\langle \text{Clickthrough}_{V,\text{TOP}}(\chi_{V,\text{TOP}}), \text{NormCTR}_{V,\text{TOP}}(\chi_{V,\text{TOP}})\rangle$ on this curve, where $\chi_{V,\text{TOP}}$ is the TOP threshold picked for the flight. The operating curve, in contrast, provides us with a more complete picture of the performance of the algorithm by reporting the engagement metrics for the entire range of operating points (thresholds).

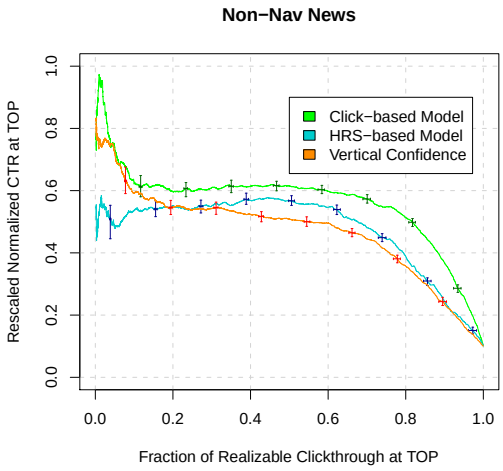

Figure 6: Normalized CTR for Non-nav News.

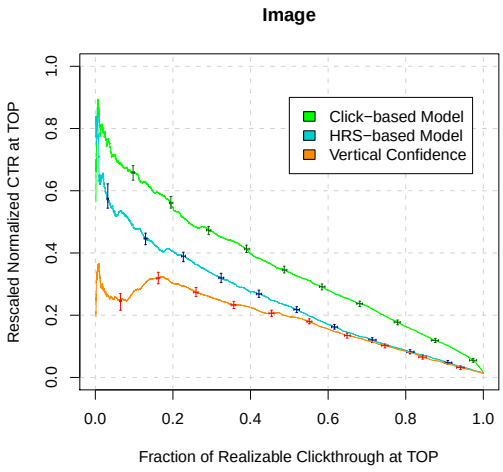

Figure 7: Normalized CTR for Image.

6.2 Sliding Normalized CTR Ratio as a means of Quality Control

We can use the approach presented in Section 4.2 to calculate the thresholds that target a certain value of a sliding quality metric. For example, we can use this to target a value α of the sliding

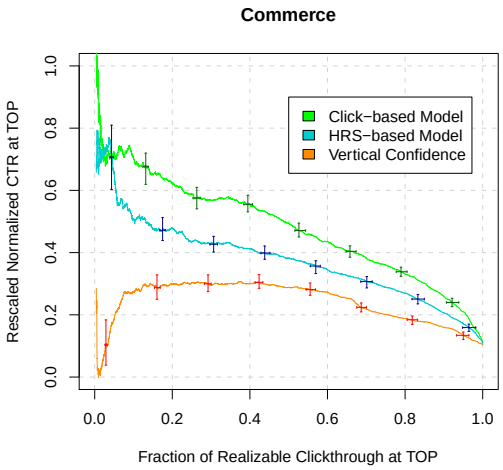

Figure 8: Normalized CTR for Commerce.

normalized CTR. Using the approach of Section 4.1, we can analyze how adjusting α changes the clickthrough to the vertical at TOP. We can in fact calculate the overall clickthrough from all slots too. But we restrict ourselves to TOP for simplicity. The trade-off curves for Image are shown in Figure 9 based on a sliding window of $w = 1000$ impressions.

In general, we expect that a better model will assign a higher score to impressions where the vertical gets a click but not the web results below it and assign a lower score to queries whose impressions have a click on the web result below the vertical, but not on the vertical itself. Thus we expect a better model to achieve a higher value of sliding normalized CTR at the highest ranges of it scores while at the same time achieve a lower value of normalized CTR at the lowest ranges of the score. In particular, if χ_{Random} is a model that assigns a random number as the score for a query, its sliding normalized CTR will be a constant (equal to $\text{NormCTR}_{V,\text{TOP}}(\mathbb{I}_{\text{audition}})$, since it will place the vertical at TOP for random queries irrespective of the TOP threshold). Thus its clickthrough vs. sliding normalized CTR curve will be a flat horizontal line. The better a model is, the more the curve will be rotated clockwise.

Also of particular interest is the graph with clickthrough at TOP on the X-axis and the normalized CTR and sliding normalized CTR at TOP on the Y-axis. For a given value α of for the sliding normalized CTR threshold, this graph allows us to read out the clickthrough that we will obtain and also the normalized CTR, the collective quality of the impressions for which we let the vertical at TOP. This graph is presented for the click-based model for Image in Figure 10.

6.3 Impact of Features

A common problem that we face is quantifying the impact of adding new features to the modeling process. For example, before we spend the resources to implement and deploy a new feature, we would like to quantify how much will it help improve our predictions. We can look at the difference in regression errors reported during the modeling process when we train models with and without the new features. But there is no natural way to map this to changes in flight metrics. In this case, we can use our approach to predict the impact on the metrics offline.

In our feature vector, we have as features the scores of various vertical classifiers. For example, when a query is received, the the

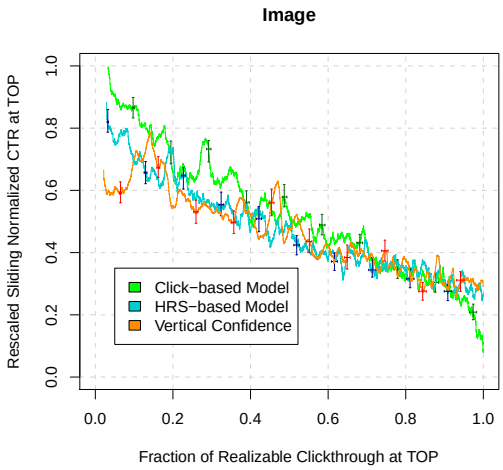

Figure 9: Sliding normalized CTR for Image.

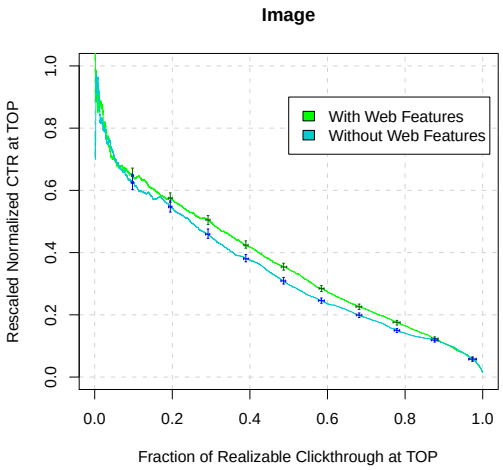

Figure 11: Dropping vertical classifier features degrades the clickthrough vs. normalized CTR trade-off.

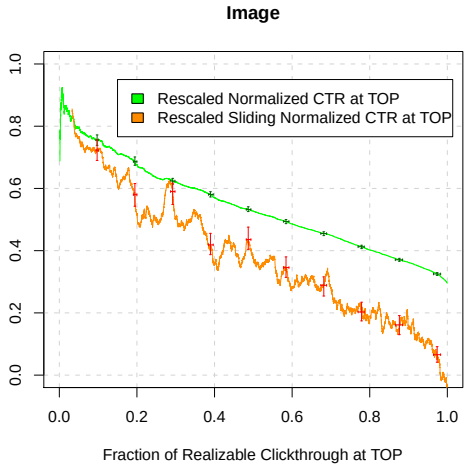

Figure 10: Normalized CTR and sliding normalized CTR for Image as a function of clickthrough.

Image classifier returns the probability that the query has image intent. We also have classifiers for some of the most important verticals. In Figure 11 we report the impact of dropping all the vertical classifier scores from the training process on the Image vertical when the model was trained on click data. In the case of Image, the vertical confidence happens to be the same as the Image classifier (and hence we dropped the vertical confidence too when we dropped the classifier scores) and is the most important feature when used. The plot shows that dropping these features results in a statistically significant drop in the performance of the model.

6.4 Impact of Head versus Tail Weights

When training rankers for vertical search results using clicks as labels, one needs to make sure that there is a fair representation of head, body and tail queries in the training data. The frequency distribution of queries that trigger results from each vertical varies considerably across verticals and hence it is important to characterize this distribution of queries in the training data for each vertical individually. Furthermore, it is important to find the relative representativeness (or weighting) of head and tail queries in the training data that would optimize some user engagement metric such as the normalized CTR at TOP across specific query segments as well as across all queries. The model operating curves described earlier gives us a straight forward methodology to choose an optimal weighting combination for head and tail queries.

We typically collect query impressions from over a two week time-frame on the auditioning flight (referred to in Section 5.2) to be able to train a ranker for a vertical. We assign each impression in this training data the normalized logarithmic query frequency over the 2 weeks as the base weight. In order to perform a sweep of possible weighting combinations across head and tail queries, we assign the head queries (with greater than 10 query impressions in 2 weeks) a weight of α and the tail queries (the remaining queries) a weight of $(1-\alpha)$. For each α value, we pick an optimal model after comprehensive parameter sweeps during the training process. We then plot the model operating curves for the models corresponding to each α value between 0 and 1. These curves use query impressions from an evaluation set collected from a two week time-frame in the future on the same flight from which the training impressions were obtained. Furthermore, we could also filter these impressions into head and tail and plot separate model operating curves for the two segments of the evaluation set to see how the weighting impacts performance in each segment.

We performed the head versus tail weighting experiments for three verticals — News, Commerce and Image. Figure 12 shows the model operating curves of a Non-Navigational News ranker for head weights of $\alpha = (0.0, 0.2, 0.4, 0.6, 0.8, 1.0)$ on all query impressions in the evaluation set. Figures 13 and 14 show the operating curves on the head and tail evaluation segments respectively. As can be seen, the value of α for which we get the optimal model operating curve over most of the clickthrough ranges is 0.2 overall. At $\alpha = 0.2$, the base weights (logarithmic query frequency) for head queries are assigned a weight of 0.2 and the tail queries are assigned a weight of 0.8, which happens to make the tail impressions $4X$ as representative as it was originally relative to the head impressions. Interestingly, this is optimal even on the head segment beating a model trained only on head queries (which is represented by the operative curve for the head weight of $\alpha = 1.0$) indicating the incremental value of having tail impressions to train a model for ranking head queries. As for the tail segment, the operating

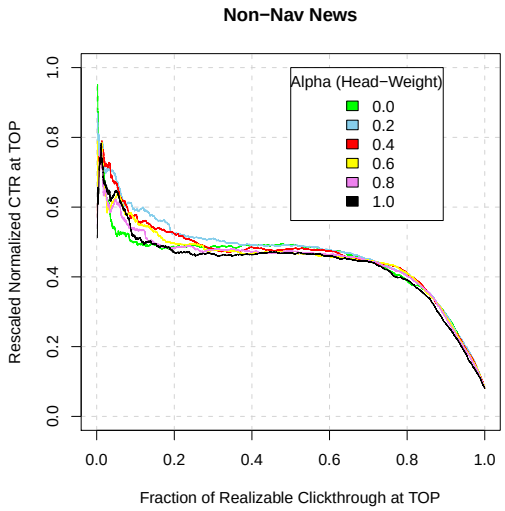

Figure 12: Normalized CTR for Non-nav News for head weights between 0 and 1.

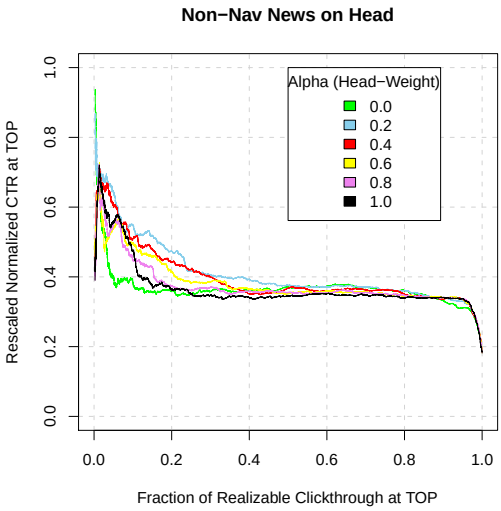

Figure 13: Normalized CTR for Non-nav News on head queries for head weights between 0 and 1.

curve for $\alpha = 0.2$ runs neck and neck with the curve for $\alpha = 0.0$ (training only on tail impressions) and does better at certain operating points on the curve, indicating the incremental value of using head query impressions when training a ranker for tail queries if one wishes to operate at these points on the curve. The operating curves for the other two answers, Commerce and Image (not shown in the paper), also showed "sweet-spots" over distinct α values giving the modeler a choice of head and tail weights he or she could use depending on what segment of queries he or she wants to cater to or what user engagement metric he or she is shooting for.

6.5 Impact of Modeling Loss Function

Candidate models (or the predictions they make) can only be compared once a loss function is decided upon. But what evidence is used to select the best loss function for the problem domain? One can conduct time consuming online experiments, but here we show a characterization curve can be used to compare the different options offline without experimenting on real users.

To demonstrate this, we trained two models models using different loss functions and used a characterization curve to compare them. One model used the L2 loss function and the other used a logistic loss function. The training process used was described in Section 5.2 including a parameter sweep. The click labels were generated from the process described in Section 5.2.

Operating curves were generated for both models using impressions from a different 2 week time-frame of the auditioning flight. These curves are shown in Figure 15 and we can see that although the model trained with L2 appears to be marginally better than that trained with logistic loss, the difference is not statistically significant. Thus, the methodology allows us to weed out such hypotheses without conducting an online experiment.

6.6 Offline Predictions Vs. True Metrics for Flights

To validate that the assumptions we made in Proposition 1 can be used to make meaningful predictions, we compared some of the predictions to true flight metrics. We used the click-based model for Commerce, and set up two flights with two different TOP thresholds. We used Proposition 1 to predict the clickthrough and normalized CTR for Commerce at TOP. The relative differences between the predicted and observed values of clickthrough on the two flights were -8.6% and -3.6%. The negative values indicate that the observed values were lower than the predicted values. The relative differences between the predicted and observed values of normalized CTR on the two flights were -3.2% and -6.8%.

7. CONCLUSIONS

We showed how we can use impressions from an auditioning flight to perform many offline experiments to study the impact of system operating point changes on click metrics. The characterization curves give an comprehensive picture of the federated web search system over the entire range of operating points and on at one specific operating point like online experiments typically do. In addition the characterization curves can be generated offline and so are not time-consuming like the online experiments. The results in Section 6.6 show that the offline predictions we make are reasonably accurate.

We demonstrated the methodology by applying it to answer various questions about our federated web search system. We used it to understand the trade-off of clickthrough and quality, impact of specific features on clickthrough, impact of various query sampling strategies on user engagement metrics, and, finally, the impact of modeling loss functions on engagement metrics.

One improvement we can do to our auditioning flight is that we can reduce the amount of auditioning needed for head queries. For some of the most common queries, we can stop auditioning verticals once we hit a certain number of impressions.

Although our approach works for a large number of natural metrics that we look at, it can not be used to predict session based metrics. It works well only for any SERP level vertical metrics. Also we won't be able to accurately predict metrics for low-coverage verticals since they have very few impressions.

8. REFERENCES

[1] J. Arguello, J. Callan, F. Diaz, and J. F. Crespo. Source of evidence for vertical selection. In *Proc. of Ann. Intl. ACM SIGIR Conf. on Research and Development in Information Retrieval*, 2009.

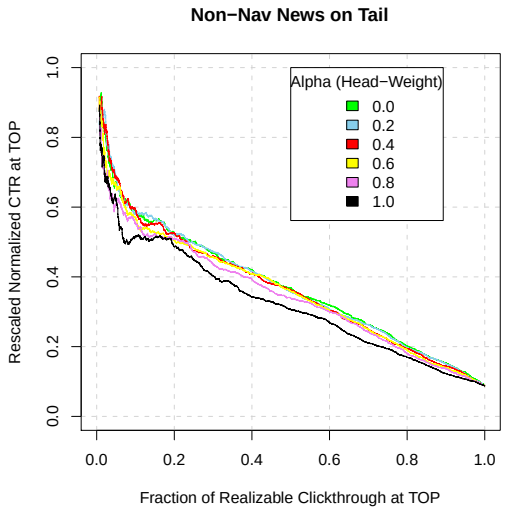

Figure 14: Normalized CTR for Non-nav News on tail queries for head weights between 0 and 1.

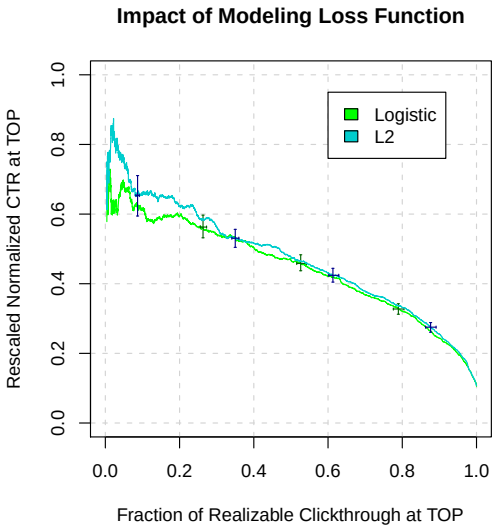

Figure 15: Normalized CTR for Commerce models trained with different loss functions.

[2] O. Chapelle and Ya Zhang. A dynamic bayesian network model for web search ranking. In *Proc. of Intl. Conf. on World Wide Web*, 2009.

[3] K. Collins-Thompson. Accounting for stability in retrieval algorithms using risk-reward curves. In *Proc. of SIGIR 2009 Workshop on the Future of Evaluation in Information Retrieval*, pages 27–28, 2009.

[4] K. Collins-Thompson. Reducing the risk of query expansion via robust constrained optimization. In *Intl. Conf. on Information and Knowledge Management*, pages 837–846, 2009.

[5] N. Craswell, O. Zoeter, M. Taylor, and B. Ramsey. An experimental comparison of click position-bias models. In *Intl. Conf. on Web Search and Data Mining*, 2008.

[6] F. Diaz and J. Arguello. Adaptation of offline selection predictions in presense of user feedback. In *Proc. of Ann. Intl. ACM SIGIR Conf. on Research and Development in Information Retrieval*, 2009.

[7] G. Dupret and C. Liao. A model to estimate intrinsic document relevance from the clickthrough logs of a web search engine. In *Intl. Conf. on Web Search and Data Mining*, 2010.

[8] B. Efron and R. J. Tibshirani. *An Introduction to the Bootstrap*. CRC Press, 1993.

[9] D. Fetterly, N. Craswell, and V. Vinay. The impact of crawl policy on web search effectiveness. In *Ann. Intl. ACM SIGIR Conf. on Research and Development in Information Retrieval*, pages 58–587, 2009.

[10] J. H. Friedman. Greedy function approximation: A graidient boosting machine. *Annals of Statistics*, 29:1189–1232, 2001.

[11] K. Fukunaga. *Statistical Pattern Recognition*. Academic Press, 1990.

[12] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning*. Sringer-Verlag, New York, NY, 2001.

[13] D. Hawking, N. Craswell, P. Thistiewaite, and D. Harman. Results and challenges in web search evaluation. *Computer Networks*, 31(11-16), 1999.

[14] S. Ji, T. Moon, G. Dupret, C. Liao, and Z. Zheng. User behavior driven ranking without editorial judgments. In *Proc. of Intl. Conf. on Information and Knowledge Management*, 2010.

[15] T. Joachims. A support vector method for multivariate performance measures. In *Proc. Intl. Conference on Machine Learning*, 2005.

[16] T. Kanungo, M. Y. Jaisimha, J. Palmer, and R. M. Haralick. A quantitative methodology for analyzing the performance of detection algorithm. In *Proc. of IEEE International Conference on Computer Vision*, pages 247–252, 1993.

[17] N. A. Macmillan and C. D. Creelman. *Detection Theory*. Lawerence Earlbaum Associates, Inc., 2005.

[18] A. K. Ponnuswami, K. Pattabiraman, Q. Wu, R. Gilad-Bachrach, and T. Kanungo. On composition of a federated web search result page: Using online users to provide pairwise preference for heterogeneous verticals. In *Proc. of Intl. Conf. on Web Search and Data Mining*, 2011.

[19] F. Radlinksi and T. Joachims. Minimally invasive randomization for collecting unbiased preferences from clickthrough logs. In *Proc. of AAAI Conf. on Artificial Intelligence*, 2006.

[20] F. Radlinski and T. Joachims. Query chains: Learning to rank from implicit feedback. pages 239–248, 2005.

[21] M. Shokouhi and L. Si. Federated information retrieval. In D. W. Oard and Editors F. Sebastiani, editors, *Foundations and Trends in Information Retrieval*. 2010.

[22] L. Si and J. Callan. Modeling search engine effectiveness for federated search. In *Proc. of Ann. Intl. ACM SIGIR Conf. on Research and Development in Information Retrieval*, 2005.

[23] H. L. Van Trees. *Detection, Estimation, and Modulation Theory, Part 1*. John Wiley and Sons, Inc., 2001.

[24] E. M. Voorhees and D. K. Harman, editors. *TREC: Experiment and Evaluation in Information Retrieval*. MIT Press, 2005.

[25] K. Wang, N. Gloy, and X. Li. Inferring search behaviors using partially observable Markov (pom) model. In *Intl. Conf. on Web Search and Data Mining*, 2010.