On Composition of a Federated Web Search Result Page: Using Online Users to Provide Pairwise Preference for Heterogeneous Verticals

Ashok Kumar Ponnuswami
Microsoft Corporation
One Microsoft Way
Redmond, WA 98052

Kumaresh Pattabiraman
Microsoft Corporation
One Microsoft Way
Redmond, WA 98052

Qiang Wu
Microsoft Corporation
One Microsoft Way
Redmond, WA 98052

Ran Gilad-Bachrach
Microsoft Corporation
One Microsoft Way
Redmond, WA 98052

Tapas Kanungo
Microsoft Corporation
One Microsoft Way
Redmond, WA 98052

ABSTRACT

Modern web search engines are federated — a user query is sent to the numerous specialized search engines called *verticals* like web (text documents), News, Image, Video, etc. and the results returned by these engines are then aggregated and composed into a search result page (SERP) and presented to the user. For a specific query, multiple verticals could be relevant, which makes the placement of these vertical results within blocks of textual web results challenging: how do we represent, assess, and compare the relevance of these *heterogeneous* entities?

In this paper we present a machine-learning framework for SERP composition in the presence of multiple relevant verticals. First, instead of using the traditional label generation method of human judgment guidelines and trained judges, we use a randomized online auditioning system that allows us to evaluate triples of the form <query, web block, vertical>. We use a pairwise click preference to evaluate whether the web block or the vertical block had a better users' engagement. Next, we use a hinged feature vector that contains features from the web block to create a common reference frame and augment it with features representing the specific vertical judged by the user. A gradient boosted decision tree is then learned from the training data. For the final composition of the SERP, we place a vertical result at a slot if the score is higher than a computed threshold. The thresholds are algorithmically determined to guarantee specific coverage for verticals at each slot.

We use correlation of clicks as our offline metric and show that click-preference target has a better correlation than human judgments based models. Furthermore, on online tests for News and Image verticals we show higher user engagement for both head and tail queries.

Categories and Subject Descriptors

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

WSDM'11, February 9–12, 2011, Hong Kong, China.

General Terms

Algorithms, Experimentation, Human Factors, Measurement

Keywords

federated web search, heterogeneous verticals, pairwise preference from clicks, randomized flights, machine learning

1. INTRODUCTION

The core of web search is displaying links to relevant web pages for a given query. However, modern web search engines, have access to several alternative data sources and it is now the expectation of users to have a blend of multiple types of information in the search result page (SERP). For example, many users who type the query "cool cars" are interested in seeing images of cars, whereas users who type the query "weather" are interested in the weather forecast at their current location. Therefore, web search engines are federated search engines (see, e.g., [21]) in the sense that they have multiple data sources and they need to rank the relevancy of different types of data items and display them accordingly. In many cases these different data sources are referred to as verticals [6, 2, 1] and the challenge could be presented as identifying and ranking the relevancy of different verticals given a query.

The main challenge we address is the problem of page composition. Given the different types of information, the goal is to layout the SERP with the most relevant information. In a sense, each vertical provides few data blocks. These blocks compete on their location on the SERP. Since users are more likely to see and interact with items presented at the top of the page, it is essential to rank the different information blocks and place them on the page accordingly.

When viewed as a ranking problem, there is a big difference between the core ranking problem and the federated search type of ranking problem. In the core ranking problem one has to compare object of the same nature, i.e. web pages, and rank them according to their relevancy. However, in the federated search problem, one has to compare objects of different nature, e.g. web pages to images, and value their relevancy. The different object cannot be represented using similar features to allow them to be ranked using the same machinery used to rank web documents. For example, web pages have properties such as number of outgoing links, BM25 [18] and so on. However, these features do not exist in images. At the same time, images have properties such as size, color palette and

other features which do not have natural equivalent in the world of web documents. Therefore, the main challenge we are interested at is how to merge, or integrate, heterogeneous results.

Another problem in federated search comes from the process of collecting judgments to be used for choosing, or training, a ranking model. The common method to collect such labels is to hire human judges to score each query-result pair. Recently, a lot of attention was given to obtaining labels by analyzing users click logs (e.g., [4]). The problem is that when the objects are of different type, it is hard to calibrate the scales of the scores to generate a unified scale. For example, can you compare an excellent picture of dogs to an excellent web document about dogs?

To overcome these challenges, we propose a new point of view on the federated search problem in web search. Instead of trying to associate an absolute score to each vertical, we measure the relevancy in comparison to the core web results. Since the core web results are the pivot around which the rest of the SERP results are presented, we use it as an anchor. The question we are trying to solve now becomes: given a set of links to web pages and a set of say, images, should the images be presented above the web results, bellow the web results or not at all. When collecting judgments, we present the two blocks, web results and vertical results, and ask judges to quantify the *relative* quality of the two blocks. In the same way, when ranking, we always have a web block and an vertical block coming from an alternative source. We have a unified representation for all query-web block-vertical triples which allows us to use machine learning technique to layout the SERP. Therefore, this novel point of view solves the two main challenges discussed above.

In this paper we present a detailed explanation of our solution as well as the results of experimenting with it in one of the leading web search services. We show that this solution significantly improves over previous propositions. We also show how explicit feedback, from human judges, and implicit feedback, from clicks, can be provided when learning these models to further improve the results.

2. RELATED WORK

Several research projects addressed related problems to the one discussed here. In this section we present this prior art and discuss the commonalities and differences.

2.1 Federated Information Retrieval

Federated information retrieval (FIR) addresses the problem of searching multiple *text* collections simultaneously and returning a set of merged results [19, 20, 21, 16].

In this line of work, it is assumed that the entities in each collection are similar (text documents) and these entities can be described using similar features and attributes. The focus areas here are i) what to do when you do not have access to all the text collection documents but can only query them, ii) how do you select the right collection, iii) how to integrate results from multiple web search engines. In these scenarios, it is assumed that information retrieval features like BM25 can be used to describe entities in each of collection.

However, in our problem, we would like to combine the best results from multiple specialized search engines, each of which returns best results from collections containing very different types of entities such as images, videos, news, local business listings, etc. Thus we are faced with selecting and merging results that are heterogeneous and in general cannot be described by the usual text features.

2.2 Learning to Rank

Using machine learning techniques [10] to learn relevance models for text documents has been an active area of research. Burges *et al.* used machine learning to rank documents [15, 7], Zheng *et al* used stochastic gradient boosting [9] for web ranking [23]

In the above approaches the entities that are ranked are of the same type (text). Since we have heterogeneous types of entities, we would need guidelines to compare different types of entities, and some sort of absolute measure of goodness across different entities with different visual presentation (e.g. text versus images versus video).

Pairwise preference judgments have been shown to give a better ranking performance in terms of text retrieval measures [3, 22]. In these approaches, it is again assumed that the entities are similar can be represented using similar features.

In the above approaches, the target label that the optimization algorithm tried to learn was created by trained human judges who were provided with specific guidelines for grading documents. The drawback of this framework is that label generation process can be laborious, time consuming, and expensive. Furthermore, in many situations (e.g., if query has a local intent and the judge is not familiar with the user's city) it is difficult for the judge to know the intent of the user and hence can be incorrect in creating the label. This motivated researchers to seek label generation processes that are based on past users interactions.

Joachims used clicks and skips as preference judgments for learn ranking models [12]. Chapelle and Zhang [4] showed that adding pairwise click preference as labels to human judgment data can improve NDCG when comparing results against human judgments. Ji *et al.* [11] used click-as-target to learn ranking models for ranking entities *within* a specific (Local) vertical search collection. Joachims and Radlinski [17] showed how one can randomize consecutive search result pairs and use the engagement difference to create preference labels. Similar to this line of work, we too use user engagement as a source for labels to train learning to rank models. However, the problem we face is slightly different. The main difference is that since we need to compare objects of different types, we use the historic data to provide pair-wise preferences among heterogeneous entities as opposed to previous techniques which compared similar textual results.

2.3 Web Vertical Selection

In web search, *verticals* are specialized search engines for: images, videos, local business listings, news, shopping, etc. Diaz *et al.* have published a series of papers on this topic and these papers are the closest in spirit to the work presented in this paper. However, these papers are mostly focused on the problem of triggering, which is how we select the best relevant vertical. In our work, we address the problem of composition, which is given *multiple*, already known to be relevant verticals, how does one place them relative to web blocks.

Diaz [5] addressed the problem of whether or not to show a specific item for a given query above the web results. They tried to estimate the click-through rate of the news item by learning a model based on the frequency of the query in the vertical within a specific time interval, and the frequency in the main web vertical, and non-document features. One of the issues with this approach is that estimating click-through is not possible for the tail directly. Second, the web document specific and vertical specific features, which have significant information about the specific document and news items that are being considered is not taken into the picture. Furthermore the click-through rate is very prone to presentation bias, which has a major impact on the quality of the results. Finally, the work presented is for placing news at only one location on the page, how one will present it at other locations is not addressed.

Agruello *et al* [1] present an approach where human judges are used to generate ground truth labels as to which verticals (up to six of eighteen verticals) are relevant. Then various machine learning algorithm were trained using query log, query, and corpus features to score the verticals. In each of these cases, the web result itself

is not factored in and so a vertical's selection is independent of the quality of the core web result. In addition, it is assumed that guidelines can be created to specify which vertical is the intent of a user for a specific query, which is questionable. In a more recent work [6] the authors have presented an algorithm that uses clicks to update the base model by estimating click-through. This work suffers from the fact that the approach, as presented, can only be used for updating head query vertical selection.

Finally, Agruello *et al* [2] argue that we can *port* models trained for vertical with training data (human judgments) to other verticals for which one does not yet have any human judgment data.

In the above algorithms, there is an assumption that judgment guidelines and human judges can recognize the intent of the user while issuing a query. This assumption is questionable – what if the query from a remote city about some local landmark and the judge has no knowledge of that city? The papers also don't explicitly address the issue of comparing a vertical's relevance in the context of the web results. The issue of how to place the verticals at lower ranks on the page is also not addressed. Finally, the adaptation using clicks does not address how to update the model for tail queries.

3. THE PROBLEM STATEMENT

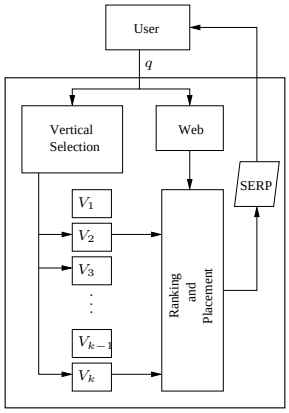

Figure 1: A high level overview of how the SERP is generated in response to a user query.

We start with a brief description of how the SERP is generated concentrating on the steps relevant for vertical ranking. When a user query q arrives at Bing, it is passed on to various components that generate parts of the SERP. This is summarized in Figure 1. One component generates the web (URLs for text documents) results that are displayed. In parallel, another module does a preliminary check on what verticals might be relevant and passes the query to the components corresponding to those verticals. This step is called *triggering* or *vertical selection*. In our example, V_2, V_3, and V_k have triggered. Some subset of those verticals might generate content that are then passed to a central ranking and placement algorithm along with the web results. In the example, vertical V_1 did not have any relevant content and was suppressed.

The ranking and placement component then generates the SERP and returns it back to the user. The focus of our work is on the ranking and placement component. Most of the related literature focuses on the triggering component.

Let us say a user issued a query q to the search engine. Let the set of web results that are selected for this query be denoted by W_q. Let the verticals that are triggered for the query be $V_1, V_2, \ldots, V_m$. Suppose that the slots that the verticals can be displayed at are $S = (s_1, s_2, \ldots, s_k)$ as shown in Figure 2. The problem we are trying to solve is to come up with a function f_V that given a query q, the web results W_q and a vertical V with content v_q, returns the slot s

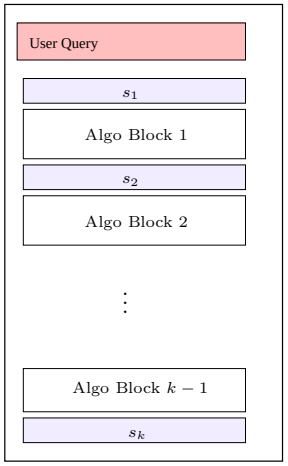

Figure 2: A simplified version of the SERP showing slots where the verticals can be placed. A fixed number of web results can be placed between two consecutive slots.

at which the vertical should be displayed, that is,

$$s = f_V(q, W_q, v_q).$$

We require the function f_V to satisfy some coverage constraints for vertical V. Suppose we want to ensure that the vertical V is required to be shown c_i fraction of the time at position s_i when the vertical triggers. Then for a randomly sampled set Q of queries for which V triggers, we require that

$$|\{q \in Q : f_V(q, W_q, v_q) = s_i\}|/|Q| = c_i \; \forall i = 1, 2, \ldots k.$$

In our case, we have $k = 3$ slots where we can place the vertical. The slots s_1, s_2 and s_3 are commonly referred to as the top of page (ToP), middle of page (MoP) and bottom of page (BoP). Instead of the labels s_i, we will simply refer to these slots by their abbreviations for clarity. Subject to the coverage constraint, we would like to find a ranking function f_V that places the vertical at ToP for c_{ToP} fraction of the queries for which the vertical is most relevant, the next c_{MoP} fraction of queries for which the vertical might be relevant at MoP and finally place the vertical at BoP for the remaining queries. Note that the coverage requirements are defined per query impression and not just by the number of unique queries. Therefore, if the same query is issued ten times, and the model displays the vertical V_i at ToP for this query, we count this as ten queries when computing the coverage c_{ToP}.

To determine the relevance of the vertical for the query, we use the click-through rate (CTR) as a measure. We want the function f_V to improve the click-through of the vertical at ToP compared to the existing production function.

4. OUR APPROACH

We solve the SERP composition problem in two stages:i) Ranking and ii) Calibration and Placement.

4.1 Ranking

We first represent the triple (q, W_q, v_q) corresponding to every query q by a *feature vector* $\phi_V(q, W_q, v_q) \in R^d$. This will comprise of values that represent the relevance of the web results and the vertical to the query. Let us say a query q is issued by a user to the search engine and the vertical V is triggered for q. We can immediately compute $x = \phi_V(q, W_q, v_q)$ from it. Using x, we would like to compute one value that reflects the relevance of the of the vertical for the query. That is, we want to find a function χ_V such that $\chi_V(x_1) \geq \chi_V(x_2)$ implies that the vertical will be placed at least as high on the page for query q_1 as for query q_2,

Figure 3: A screenshot of the SERP showing results from verticals at ToP, MoP and BoP.

where x_1 and x_2 are the feature vectors for the queries q_1 and q_2 respectively. To find this function χ_V, we generate a set of pairs $\langle x, l \rangle$ that we call examples. Then we try to find a function χ_V that best fits the data. For this, we use a learning algorithm, which is described in Section 5.

In an example $\langle x, l \rangle$, l is referred to as a *label*. In some sense, the l is the value we want the function χ_V to predict for input x. We can provide contradicting examples too. In this case, the learning algorithm tries to find χ_V that best fits these examples under some metric. We may also associate a weight w with each example $\langle x, l \rangle$ that indicates the relative importance of the example. So our input to the learning algorithm is a list of tuples $\langle x, l, w \rangle$ There are two common methods for generating the examples for training:

- Sample a set of queries for which the vertical triggers. Then show the contents of the web results and the vertical to judges and ask them to assign a label, say 0 or 1 for relevant or not relevant, or a number in some range like $0, 1, 2, 3$ indicating the degree of the relevance of the vertical for the query.

- Sample a set of queries for which the vertical triggers from the user logs, and based on the user behavior, assign a label, say 0 or 1 to indicate if the vertical may have satisfied the user's information need.

In this paper, we concentrate on the latter approach using the former only for comparison in our offline analysis. The details of our label generation process follow in Section 5.3.

4.2 Calibration and Placement

Once we arrive at a ranking function r_V, our next task is to match coverages at ToP, MoP and BoP based on our agreements with vertical owners. This is a straight-forward process. Given a sample Q of queries for which the vertical triggers, we compute the value provided by the ranking function χ_V for these queries, sort them and pick two thresholds $\chi_{V,ToP}$ and $\chi_{V,MoP}$ such that

$$|\{q \in Q : \chi_V(\phi(q, W_q, v_q)) \geq \chi_{V,ToP}\}|/|Q| = c_{ToP}$$

and

$$|\{q \in Q : \chi_{V,ToP} > \chi_V(\phi(q, W_q, v_q)) \geq \chi_{V,MoP}\}|/|Q| = c_{MoP}.$$

This gives us a function $\mu_V : \mathbb{R} \longrightarrow S$ where

$$\mu_V(x) = \begin{cases} ToP & \text{if } x \geq \chi_{V,ToP} \\ MoP & \text{if } \chi_{V,ToP} > x \geq \chi_{V,MoP} \\ BoP & \text{otherwise.} \end{cases}$$

Finally, the function f_V that we wanted to compute can be written as

$$f_V(q, W_q, v_q) = \mu_V(\chi_V(\phi_V(q, W_q, v_q))).$$

For convenience, with a slight abuse of notation, we will drop the vertical V from the subscript (since we rank each vertical individually) and we may use $\chi(q)$ to denote $\chi_V(\phi_V(q, W_q, v_q))$.

5. STATISTICAL MODELING

In this section, we first describe Gradient Boosted Decision Trees (GBDT), which is the machine learning algorithm we used to learn online user preference and predict human relevance grades. Then we describe how we represent the data using features such that the web component of the feature vector becomes a reference frame or anchor, and finally we describe how we generate training labels from online user click logs.

5.1 Gradient Boosted Decision Trees

Gradient boosted decision trees (GBDT) are non-parametric regression models [8]. GBDTs and its stochastic variant [9] compute a function approximation by performing a numerical optimization in the function space instead of the parameter space. We provide an overview of the GBDT algorithm.

A basic regression tree $f(x), x \in \mathcal{R}^K$, partitions the space of explanatory variable values into disjoint regions $R_j, j = 1, 2, \ldots, J$ associated with the terminal nodes of the tree. Each region is assigned a value ϕ_j such that $f(x) = \phi_j$ if $x \in R_j$. Thus the complete tree is represented as:

$$T(x; \Theta) = \sum_{j=1}^{J} \phi_j I(x \in R_j),$$

where $\Theta = \{R_j, \phi_j\}_1^J$, and I is the indicator function. Let the given training data be denoted by $(x_i, y_i, w_i), i = 1, \ldots, N$. That is, x_i are the observed feature vectors, y_i the labels, and w_i the weight associated with the training pair (x_i, y_i). For a loss function $L(y_i, \phi_j)$, parameters are estimated by minimizing the total loss:

$$\hat{\Theta} = \arg\min_{\Theta} \sum_{j=1}^{J} \sum_{x_i \in R_j} w_i \cdot L(y_i, \phi_j).$$

In our experiments, we perform regression using the squared error as loss function.

A boosted tree is an aggregate of such trees, each of which is computed in a sequence of stages. That is,

$$f_M(x) = \sum_{m=1}^{M} T(x; \Theta_m),$$

where at each stage m, Θ_m is estimated to fit the *residuals* from the $m-1$th stage:

$$\hat{\Theta}_m = \arg\min_{\Theta_m} \sum_{i=1}^{N} w_i \cdot L(y_i, \eta f_{m-1}(x_i) + \phi_{j_m}).$$

In practice, instead of adding $f_m(x)$ at the mth stage, one typically adds $\eta f_m(x)$ where η is a the *learning rate*. This is similar to a "line search" where one moves in the direction of the gradient, but the step size need not be equal to the gradient.

In the *stochastic* version of GBDT, instead of using the entire data set to compute the loss function, one sub-samples the data and finds the values ϕ_j that minimize the loss on the test set. The stochastic variant minimizes over-fitting. The depth of the trees in each stage is another algorithm parameter of importance. Interestingly, making the trees in each stage very shallow while increasing the number of boosted trees tends to yield good function approximations. In fact, even with depth 1 trees, often called stubs, it possible to achieve good results. Interaction amongst explanatory variables is modeled by trees of depth greater than 1.

In practice, one has to empirically set (by cross-validation) the parameters: the number of trees M, the number of nodes per tree P, (which is related to J), learning rate η, and sampling rate ρ, (in the stochastic version).

5.2 The Feature Vector

When presenting the implicit user feedback data to the machine learning package, we need a way to jointly capture the relevance of the vertical *and* the web results for each query. To achieve this, we create a *hinged feature vector* $x = \phi(q, W_q, v_q)$ that is composed of features that correspond to both the vertical and the web results. Thus for a query q if three verticals V_1, V_2, and V_3 trigger, each of them is paired with the same web block and so the feature vector all the three verticals have identical web components, but different vertical component. That is, the web part of the feature vector creates a reference frame or an anchor against which the heterogeneous verticals are scored. We briefly describe the features we use in our modeling process. These can largely be broken down into the following categories:

1. *Query-Web based features:* These are features that measure how well the web results match the query. Some examples are:
   - Click-through rates (CTRs) of the web results for the query at various locations. These CTRs are available for head queries. The dwell times of the web results are also present as features.
   - The number of web results that belong to a certain set of domains. For example, one domain set consists of `http://www.wikipedia.org/` and `http://en.wikipedia.org/`. There are four other such sets. In addition to domain matches, we also have features that measure matches to some path-lists.
   - BM25Fs of the ten web results,
   - The maximum BM25F of the ten web results.
2. *Query-vertical based features:* These are features that measure how well the vertical matches the query.
   - Vertical confidence: This is usually a feature provided by the vertical partner team based on how relevant they think the vertical content might be to the query.
   - Historic click-through rate of the vertical for the query at ToP, MoP and BoP gathered over the past six months. These statistics are available for head queries. For the position ToP, there is a Boolean variable VerticalToPKnown that indicates if the historic CTR is available for that query-vertical pair. VerticalToPCTR has the CTR if this Boolean variable is true. The analogous variable for MoP and BoP are also available.
3. *Features based on the query only:*
   - Most features of this type are scores from classifiers that indicate the likelihood of the query belonging to a particular domain, like sports, news, commerce, etc.
   - IsNavQuery: This is an indicator of whether the query is considered to be of navigational intent.
   - Features that indicate if a query is spiking in the past few hours. One of them is a Boolean feature and the other measures the spiking volume.
   - Query length.
4. *Whole-page features:* These are features that measure the historic CTRs of miscellaneous components on the page (like ads, query suggestions, pagination, table of contents, etc).

5.3 Label Generation

We now describe how we convert the impressions for an vertical into training examples. We gather impressions from a *randomized flight*. A flight is a small bucket of Bing users who are exposed to new ranking experiments that we conduct in order to get online measurements of model performance. On a randomized flight, in response to a query, we *audition* the verticals that trigger for that query at random slots on the SERP. One big advantage of using impressions where the vertical is placed independent of any feature is that the model we generate is not influenced by the existing production ranking system. So although we are able to gather less data, there is also less noise from other models. Our strategy for converting the impressions to training examples can be thought of as mainly competing the vertical against the first web block (the top three web results for the query) using a *pairwise click preference*. For example:

1. If the vertical was shown at ToP, and the vertical got a click but none of the web results in the first block did, then we label this impression 1 (a victory for the vertical over the first web block).
2. If the vertical was shown at ToP, and the vertical did not get a click but one of the web results in the first block did, then we label this impression 0 (a defeat for the vertical).
3. If the vertical was shown at MoP, and the vertical got a click but none of the web results above it did, then we label this impression 1. But since the user took the extra effort to skip over the web results to click the vertical, the weight w that we give this impression is twice the weight as in Case 1. At this point we did not tune the value of this weight.
4. All other impressions are ignored. This includes cases where both the vertical and the top web block were clicked. In future, if the vertical was shown at ToP or MoP, and both the vertical and web we might consider the order of the clicks to decide who won. Currently we also drop impressions where neither the vertical nor the first web block got a click. In future, when we are working with verticals where the user need not explicitly click on the vertical satisfy their information need (like the weather or calculator verticals), we may treat abandoned page impressions (that is, impressions where nothing on the SERP got clicked) as a victory for the vertical.

The above label generation process is motivated by the eye-tracking study reported by Joachims *et al.* [13]. They report that usually users look at at least the first and second result on the page. Also, when users clicked on a result, they usually looked at all the results above it. We ignored impressions where the vertical was shown at MoP and did not get a click because the there is a pretty good chance the user did not look at it (This is especially true of verticals like News, which are hard to distinguish from the web results in a quick glance). We ignore impressions where the vertical is at BoP because there is a good chance that the user noticed neither the vertical or the web results around it. Another reason we do so is because the click-through rate of the content around BoP is very low. Hence the signal obtained from these impressions is very small.

In the end, after the training examples are generated, we split the data into train, validate and test sets by hashing the queries so that the number of queries in these buckets is roughly in the ratio 6:1:1 and then we rescale the weights of the examples as follows:

- We log-weigh the impressions of the same query. That is, if we had n impressions of a query, then the examples corresponding to the query are multiplied by $\log n/n$. This is done because we don't want some of the most frequent queries to dominate the learning process, but at the same time, we want the algorithm to pay a bit more attention to getting frequent queries right.

- The weights are again readjusted so that the head and tail queries both have same weight. This is necessary since even after the initial adjustments, the weight of the head dominates the tail. At the same time, we ensure that within each bucket, the total weight of the 0 and 1 examples are the same. This was done because most of the weight is on impressions labeled 0 and we suspected the learning algorithm might label all points 0 too. But when we performed classification with and with out this adjustment, we didn't notice any difference in the quality of the models.

6. EXPERIMENTAL PROTOCOL

In this section, we describe our experimental setup in more detail. We concentrate on two verticals, Image and News.

6.1 Data Size

Data was gathered from a *randomized flight* over a week. A flight is basically an assignment of a small set of users to a modified ranking and placement algorithm (refer to Figure 1). On a randomized flight, the verticals were shown at random slots on the page in response to user queries. The click behavior of these users was used for training our models. we had about 524,000 impressions for the News vertical at ToP or MoP and 240,000 for the Image vertical.

One of the goals while re-ranking News was that we wanted to maintain the coverage at various slots the same for navigational and non-navigational queries. If we train one machine-learned model for News, it may have changed the coverage at slots for these two categories although it maintained the coverage for News. For example, it is plausible that News does better for non-navigational queries and a combined model would on an average give higher scores for non-navigational queries compared to navigational queries, effectively increasing the ToP coverage of the former at the expense of the latter. To overcome this problem, we trained the models for the two *scenarios* separately. The number of impressions at ToP or MoP for the two scenarios were about 405,000 and 119,000.

6.2 Parameter Sweep

When using GBDTs, there are a few parameters we can tune to get close to the best possible model. For generating the model for Image vertical, we set the number of nodes permitted in the tree P to 10, 20 and 30. The maximum number of trees was set to $M = 200$. The training rate η was set to either 0.03, 0.09 or 0.15. For each possible setting of these three parameters, we built a model and picked the best model obtained using these settings. For the navigational and non-navigational News verticals, we set P to 15, 20 or 30, $M = 200$, and η to 0.03, 0.09, 0.12, 0.16 or 0.3 for the parameter sweep.

6.3 Human Judgments

For some of our offline analysis to compare the labels generated from click-based approach to the labels generated from human judges, we used a human judgment set that had already been generated for other purposes. The first step in this process is sampling queries in a way that ensures a good representation of head and tail queries. The judges are then shown the SERP with just the web results and the vertical separately. The judges are then asked to rank the vertical's relevance on a scale of 0 to 3, indicating whether the vertical should not be shown for the query, or whether it should be shown at BoP, MoP or ToP. The process is usually referred to as the human relevancy system (HRS) at Bing. For Image, navigational News and non-navigational News, we had 37258, 857 and 3406 judgments collected this way respectively.

6.4 Metrics

We report our results for a combination of some offline and online metrics that we used to evaluate the performance of our new ranking approach.

6.4.1 *Offline Metrics*

Even before we build a model from user impressions, we wanted to investigate how various features correlate to the labels we generate from the user click behavior. We used the Pearson's correlation coefficient because of the simplicity in computing it. The Pearson's coefficient of two variable X and Y is defined as $\rho_{X,Y} = \text{Cov}(X,Y)/(\sigma_X\sigma_Y)$ where σ_X denotes the standard deviation X and $\text{Cov}(X,Y)$ is the covariance of X and Y.

For a binary variable l with $\Pr[l = 1] = p$, we have $\sigma_l = \sqrt{p(1-p)}$ and

$$\begin{aligned}\text{Cov}(X,l) &= \text{E}[X\cdot l] - \text{E}[X]\text{E}[Y] = \text{E}[\mathbb{I}_{l=1}X] - p\text{E}[X]\\ &= p\text{E}[X|l=1] - p\text{E}[X]\\ &= p\text{E}[X|l=1]\\ &\quad - p(p\text{E}[X|l=1] + (1-p)\text{E}[X|l=0])\\ &= p(1-p)\text{E}[X|l=1] - p(1-p)\text{E}[X|l=0]\\ &= \sigma_l^2(\text{E}[X|l=1] - \text{E}[X|l=0])\end{aligned}$$

which gives $\rho_{X,l} = \sigma_l(\text{E}[X|l=1] - \text{E}[X|l=0])/\sigma_X$. That is, the Pearson's correlation coefficient in this case measures how well are the average values of X for the cases when $l = 0$ and $l = 1$ compared to the standard deviation of l. We would like this value to be as large as possible so that thresholding on a value of X can be used to predict l. We report values on how the features correlate to the click labels and compare these to correlation of our ranking function $\chi(\cdot)$ obtained using click labels.

6.4.2 *Online Evaluation*

To test the performance of the new ranking function f_V, we assigned a small subset of users to flights that used these ranking functions. The click-through rates for the verticals and the click-through rate of the SERP of these flights were compared to the control flight. The control flight which comprises of about 2% of Bing's traffic acts as the baseline with which every other experimental flight is compared. Every other flight comprises of about 1% of traffic and can be used to evaluate various vertical rankers and other experimental features.

7. RESULTS

In this section, we present a drill-down of how our new ranking for the Image, navigational News and non-navigational News verticals performed under some offline and online metrics.

7.1 Correlation of Features to the Click Label

Table 1: The features with highest absolute correlation on head queries to the click labels for Image vertical.

Feature name	Correlation
IsNavQuery	-0.187
SnippetConfidenceImage	0.181
WebDwell100Pos1	-0.176
VertClassifierConfidenceVideo	-0.174
WebDwell30Pos1	-0.155
VerticalToPCTR	0.147
VerticalToPLastClick	0.147
VertClassifierConfidenceImage	0.145
SnippetConfidenceVideo	-0.137
AvgNumResultsPerImpression	-0.131

Table 2: The features with highest absolute correlation on tail queries to the click labels for Image vertical.

Feature name	Correlation
VertClassifierConfidenceVideo	-0.250
VertClassifierConfidenceImage	0.224
VerticalConfidence	0.218
SnippetConfidenceImage	0.215
VerticalToPLastClick	0.180
VerticalToPCTR	0.180
WebDwell100Pos1	-0.174
WebDwell30Pos1	-0.166
PathList2Matches	0.147
SnippetConfidenceVideo	-0.134

Table 3: The features with highest absolute correlation on head queries to the click labels for non-navigational News vertical.

Feature name	Correlation
AvgNumResultsPerImpression	-0.325
WebCTR	-0.322
VerticalConfidence	0.320
WholePageCTR	-0.312
AvgWebClicksPerImpression	-0.288
SnippetConfidenceImage	0.280
VerticalMoPKnown	-0.280
WebDwell100Pos2	-0.274
WebDwell100Pos1	-0.271
WebCTRPos3	-0.267

We report the features that are most correlated to the label l for the Image and non-navigational News verticals for head and tail queries in Tables 1-4. Note that the weakest correlations are for the Image vertical on head queries while the strongest correlations are for non-navigational News on head queries.

We briefly describe some of the features in these tables. VertClassifierConfidenceX and SnippetConfidenceX are scores from classifiers that use the query text and web snippets respectively to classify the query as having intent of vertical X. The prefixes Vertical and Web describe statistics gathered for the vertical and web results for the query q (available for head queries). Suffixes like Dwell30, LastClick, CTR describe statistics gathered for the $\langle$query, vertical$\rangle$ pair. Dwell30 is the fraction of times the user clicked on the vertical and spent at least 30 seconds on the resulting page. LastClick is when the vertical was the last component clicked on the SERP. The features with ToP/MoP/BoP in them describe that the feature contains statistic of the vertical gathered at that slot for the query. For example, VerticalToPCTR means the CTR for the $\langle$query, vertical$\rangle$ pair when the vertical was shown at ToP. Similarly, for web results, the suffix PosX describes statistics for the web result at position X for the query.

We would like to point out that the vertical CTRs are not identical to the labels we use in the training process. The label depends on whether or not the vertical *and* the web block got a click. Also, the vertical CTRs are available only for queries that have been seen frequently over the past six months. These are mostly head queries. For example, for Image, these statistics are available for 12.1% (and 27.6%, respectively) of the queries we saw exactly once (and more than once, respectively) during the two week period from which we gathered impressions. As we will see later (Table 5), our improvements on tail queries are comparable to those on head, showing that our models generalize on tail queries too.

Table 4: The features with highest absolute correlation on tail queries to the click labels for non-navigational News vertical.

Feature name	Correlation
AvgNumResultsPerImpression	-0.213
VerticalMoPKnown	-0.201
WebCTR	-0.199
WebDwell100Pos2	-0.194
WebDwell30Pos2	-0.194
WebLastClickPos2	-0.192
WebDwell100Pos1	-0.190
WebLastClickPos1	-0.188
WholePageCTR	-0.187
WebDwell30Pos1	-0.181
WebCTRPos1	-0.175

7.2 Correlation of HRS Judgments and Click-Based Labels

We wanted to check how our click-based labels have the same general trend as HRS labels. For this purpose, we used the HRS sets described in Section 6.3. Since we used judgment sets we already had, the intersection of the query sets for which we had human judgments and click-based labels would have consisted only of head queries. Instead, we took the model trained using the click data, generated the scores χ_{Click} on the queries for which we had human judgments (we had the values of W_q and v_q for each of these queries exactly as they were shown to judges). We then looked at the score distribution for each HRS label. We noticed that different judges tend to give scores on a different scale and this tends to spread the click-based scores. So we restricted the HRS data to just judge who evaluated the most number of queries for each vertical. At this point, we had 2629, 592 and 150 HRS labels for Image, navigational News and non-navigational News queries respectively. The score distribution is shown for each HRS label in Figures 4-6. One can see a clear pattern that in general the higher HRS labels have higher average χ_{Click} for each of the verticals. Notice that for Image, the distribution of χ_{Click} is pretty much the same for the HRS labels 0 and 1. But the means for χ_{Click} for each HRS label are ordered the way expect them for most judges.

7.3 Correlation of Click-based Model Scores to the Click Label

The correlation of the click-based model scores the click-based labels are shown in Table 5 for the three verticals by head and tail queries. It can be seen that for all six combinations, we obtain improvements over the correlation of the feature that has the highest value.

Table 5: The Pearson's correlation of the scores from click-based models for each vertical to the click based label by head and tail queries.

	Head	Tail
Image	0.334	0.404
Navigational News	0.432	0.346
Non-navigational News	0.533	0.384

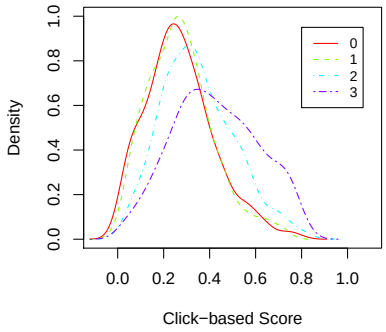

Figure 4: Distribution of scores from a click-based model for various HRS label categories for the Image vertical.

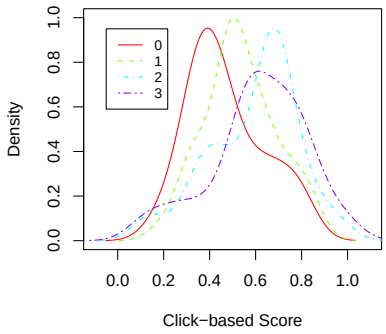

Figure 5: Distribution of scores from a click-based model for various HRS label categories for the navigational News vertical.

7.4 Correlation of HRS-based Model Scores to the Click Label

We had generated models using the HRS data set for the Image and News vertical. Due to the small size of the training set for News, we did not train separate models for the navigational and non-navigational scenarios. Let χ_{HRS} denote the ranking function obtained by modeling using HRS-based labels. The correlation of the ranking functions for the three verticals are reported by head and tail in Table 6. It is not very surprising that χ_{HRS} to l is less than that of χ_{Click}, since the click-based model was trained using click data. But it is interesting to note that the correlation lower than even the feature with even the strongest correlation to click labels in five out of the six cases.

Table 6: The Pearson's correlation of the scores from HRS-based models for each vertical to the click based label by head and tail queries. Note that a joint model was built for the two News scenarios using the data set described in 6.3.

	Head	Tail
Image	0.154	0.230
Navigational News	0.177	0.135
Non-navigational News	0.376	0.188

7.5 Feature Importance

The most important features for the Image vertical are reported in Table 7. There are significant differences in the ordering from the list of most correlated features because some of the features provide

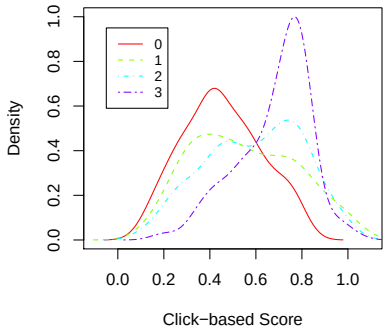

Figure 6: Distribution of scores from a click-based model for various HRS label categories for the non-navigational News vertical.

similar signals, in which case, the learning algorithm might use only one of them or use different features in different trees. For example VerticalConfidence does not appear in the Table 7 although it is has strong Pearson's correlation for both head and tail queries (0.110 and 0.218 respectively). VerticalConfidence feature turns out to be the most important feature for Image (VerticalConfidence for Image is based off VertClassifierConfidenceImage). It is the most important feature for the navigational and non-navigational News verticals too. As we see from both the feature correlations and feature importance, web and whole page features turn out to be very important at predicting clicks.

Table 7: The first 10 features in order of their feature importance in the click-based model for Image.

Feature name	Importance
VertClassifierConfidenceVideo	1
SnippetConfidenceImage	0.877
VerticalToPCTR	0.555
VertClassifierConfidenceImage	0.492
WebDwell100Pos1	0.419
IsNavQuery	0.400
SnippetConfidenceVideo	0.338
SnippetConfidenceCommerce	0.323
VerticalBoPCTR	0.322
WholePageCTR	0.297

7.6 Flight Performance

Table 8: Statistics from the Image click model flight.

Position	ΔCoverage (relative)	ΔClickthrough (relative)	ΔVerticalCTR (relative)
ToP	-10.73%	+10.6%	+23.89%
MoP	-20.40%	+5.85%	+32.98%
BoP	+17.12%	-19.89%	-31.60%

In this section, we describe the results of our online flighting experiments. All metrics reported here are relative differences between the treatment flight and the control flight that uses a hand-tuned models for ranking the three verticals. We flighted the Image model on one flight and the two News models on another flight. This was done to minimize interference between the rankers, because for a query q we can have both the Image and one of the two News verticals trigger.

We begin with a drill down for the Image flight (Table 8). While the calibration process gets us close to matching coverages at each of the ToP, MoP and BoP positions, the control flight coverages themselves have been shown to vary considerably over different days of the week and across different weeks because of user query patterns. This explains the deviation in coverages of Image ver-

Table 9: Statistics from the non-navigational News click model flight.

Position	ΔCoverage (relative)	ΔClickthrough (relative)	ΔVerticalCTR (relative)
ToP	2.35%	+3.25%	+0.88%
MoP	0.01%	-8.39%	-8.38%
BoP	-27.00%	-75.28%	-66.13%

tical in the treatment relative to the control. The third column in the table shows the relative difference in clickthrough, the absolute number of clicks to the Image vertical from the SERP. The fourth column shows the relative difference in the vertical clickthrough rate (CTR), which is the ratio of the vertical clickthrough to the total number of impressions when results from the Image vertical were shown on the SERP. Assuming we do not change the ranking function, we expect that an increase in ToP coverage would result in a increase in clickthrough but a decrease in CTR since we are showing the vertical at ToP for more, but less relevant queries. We expect the opposite trend if we decrease the ToP coverage.

As can be seen, despite the lower coverage of the Image model at ToP and MoP, the Image model gets significantly more clickthrough than control in each of these slots. In addition, we also see a significant increase in CTR at ToP (and MoP). This clearly shows the improvement of the Image model over the hand-tuned model in the control flight. It would also be worthwhile to mention here that we managed to achieve these improvements impacting the Whole Page CTR only slightly; it reduced relatively by 0.34%. The Whole Page CTR is a ratio of the number of times a click was made anywhere on the page when an Image vertical was shown to the total number of impressions when an Image vertical was shown on the SERP. It can be thought of as rate of not abandoning the page after issuing a query. With a better Image model, we expect page abandonment to increase as the thumbnails in the Image results themselves convey information not necessarily requiring a user click.

For the non-navigational News vertical too we get higher CTR at ToP (Table 9), although the improvement is less than that for Image. The CTR at both MoP at BoP decreases because ToP has higher coverage than MoP and BoP together for this vertical. So if we move better quality navigational News results to ToP, then we expect a decrease in CTR for the other two slots.

For navigational queries, we could not use the CTR as a metric because they have a strong interaction with *DCards*. DCards or definitive cards are results that are shown in response to some queries for which are very definitive on what web result most users are looking for. The DCard, if it triggers, is always the first result shown and any verticals, even if they are to be placed at ToP, are below it on the SERP. Since the DCard has a very high CTR, the other components on pages with DCard get very low CTR. Most DCards trigger in response to navigational queries and hence navigational News tends to have a high interaction with DCards. Even though we tried to match coverage at ToP to the control flight, the coverages at ToP on pages with and without DCards might have changed significantly. So for navigational news, we looked at an alternate metric. We computed the ratio of the number of times the users clicked on the vertical to the number of times the user clicked on either the vertical or the web result immediately below it. We noticed that this metric improved by 6.3% and 7.6% at ToP and MoP for both these verticals (For navigational News, MoP is the slot with highest coverage and ToP has the least. So any improvement in ranking should result in improvement of this metric for both ToP and MoP).

7.7 Training Data Size Analysis

We performed experiments to see if we collected enough user impressions to obtain the lowest possible error rate with our experimental setup. To do so, we generated samples of various sizes from

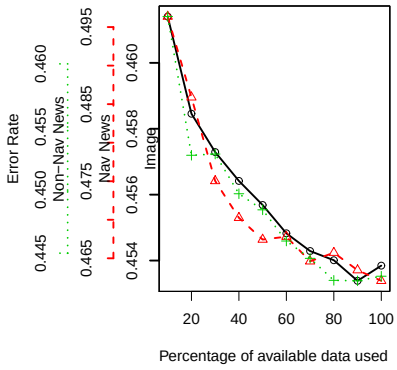

Figure 7: Training size analysis. Square root of mean-squared error as a function of the a percentage of the available data used.

the impressions we had gathered, generated a new train and validate set using these samples and measured the error rate on our original test set. The data is summarized in 7. For Image, the improvement from using 10% of the available data to 100% of the available data seems to be minimal, although there is a trend showing the error rate to be decreasing with increasing training data size for the most part. From the above analysis, it was not clear how much more improvement we could have obtained if we had gathered more data. So we gathered impressions randomized flight for navigational and non-navigational News over another week and combined it with the data we previously had. When we used 50% more data to build the models, the error rates were 0.463 and 0.443 respectively for navigational and non-navigational News. When we used twice the amount of data, the error rates were 0.461 and 0.442 respectively. The error rates were 0.462 and 0.443 respectively for the original training sets we used, showing that we were already close to the optimum error rate we could have achieved.

8. DISCUSSION

Head and Tail: Our randomized auditioning approach allows us to naturally train our models for head and tail queries. Methods that use estimates for click through rate immediately limit the scope of the modeling to head queries. Perhaps some of the query classifier features also help in smoothing the models by aggregating over tail queries in specific categories.

Clicks versus Human Judgment: In our offline experiments we systematically noticed that human judgment based models always have lower correlation with click logs. This is to be expected since the click log based models directly optimize towards clicks. However, our experiments also show that in general, verticals with higher human grade have higher model scores even in click-based models. This shows very clearly that i) training against human judgments will provide scores correlated with online user engagement, and ii) training against human judgments still leaves us a bit short. Using click logs to generate labels also helps us avoid the problem of formulating the judgment guidelines for the HRS process.

Model comparison: In our online experimental setup we found it more practical to compare the user engagement characteristics with an existing algorithm by first pegging the coverages to the engagement relative to existing composition since absolute metrics are difficult.

Image Vertical: When comparing online engagement behavior of human judgment based models and click-based models, we found that while we got much higher user engagement with Image vertical with click-based models, the overall page engagement for the

human judgment-based model was higher. Our suspicion is that the cause potentially due to the fact that many times an image might be relevant but users do not click on it, as in the case of the query {woody allen}, whereas, for the query {lady gaga } we see a lot of clicks. In the first example, the value of the vertical result is in building trust or immediate communication that the search engine "gets it."

Non-Clicky Verticals: Many verticals that do not require a click, such as Weather, can potentially introduce difficulties since there is no need for the user to click on the result. For such verticals one can either deliberately turn the vertical result clicky by forcing the users to click on a link to get the final result for a small fraction of the traffic, or creating a model based on engagement (or lack thereof) on the rest of the page [14].

Click-labels: Notice that while we consider the click as a good thing, a "satisfied" click can have different properties for different verticals. For example, the dwell-time for a satisfied click on the Image vertical can be very different from the dwell-time of the News vertical. Thus some further refinements can be done by factoring in satisfied clicks instead of just clicks.

Training Size: In our experiments we find that each vertical takes different amount of training data to "saturate" in performance. In fact, it is advisable to study the training size behavior for sub-categories of queries such as head and tail. While overall the training size might saturate, important sub-categories might still be lacking enough training data.

9. CONCLUSIONS AND FUTURE WORK

We proposed a machine learning framework for aggregating and composing results from multiple verticals with heterogeneous types of entities. The problem of creating a common relevance scale across verticals was addressed using pairwise click judgments where one of the entities in the pair is always a web result block. In feature space this was addressed by creating a hinged feature vector that consisted of two parts: a web part, and the vertical part. We showed that we can get very high correlation with user engagement clicks by building models that uses user click preference as judgments. Our models and experiments were performed using over 100k <query, vertical> pairs. Our belief is that by training models using online data we can get very high quality models, which generalize even to tail queries.

The approach presented in this paper can be adapted to address many practical problems in federated web search. Ranking and placing a new vertical result is common problem since there human training data would have to be collected to train a model. Our randomized auditioning approach easily allows us to collect training data from online users. A very similar issue appears when we try to adapt the search engine to a new international market and can be addressed using randomized auditioning.

10. 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.

[2] J. Arguello, F. Diaz, and J. F. Paiement. Vertical selection in presence of unlabeled verticals. In *Proc. of Ann. Intl. ACM SIGIR Conf. on Research and Development in Information Retrieval*, 2010.

[3] C. Burges, T. Shaked, E. Renshaw, A. Lazier, M. Deeds, N. Hamilton, and G. Hullender. Learning to rank using gradient descent. pages 89–96, 2005.

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

[5] F. Diaz. Integration of news content into web results. In *Proc. of Intl. Conf. on Web Search and Data Mining*, 2009.

[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] P. Donmez, K. M. Svore, and C. J. C. Burges. On the local optimality of LambdaRank. In *Proc. of Ann. Intl. ACM SIGIR Conf. on Research and Development in Information Retrieval*, 2009.

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

[9] J. H. Friedman. Stochastic gradient boosting. *Computational Statistics and Data Analysis*, 38:367–378, 2001.

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

[11] 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.

[12] T. Joachims. Optimizing search engines using clickthrough data. In *Proc. 8th Ann. Intl. ACM SIGKDD Conf. on Knowledge Discovery and Data Mining*, pages 133–142, 2002.

[13] T. Joachims, L. Granka, B. Pan, H. Hembrooke, F. Radlinkski, and G. Gay. Evaluating the accuracy of implicit feedback from clicks and query reformations in web search. *ACM Trans. on Information Retrieval*, 2007.

[14] J. Li, S. Huffman, and A. Tokuda. Good abandonment in mobile and PC internet search. In *PRoc. of Ann. Intl. ACM SIGIR Conf. on Research and Development in Information Retrieval*, 2009.

[15] P. Li, C. J. C. Burges, and Q. Wu. Mcrank: Learning to rank using multiple classification and gradient boosting. In *Proc. 21st Proc. of Advances in Neural Information Processing Systems*, 2007.

[16] V. Murdock and M. Lalmas. Workshop on aggregated search, 2008. `http://www.sigir.org/forum/2008D/sigirwksp/2008d_sigirforum_murdock.pdf`.

[17] F. Radlinski and T. Joachims. Minimally invasive randomization for collecting unbiased preferences from clickthrough logs. In *Proc. of AAAI*, 2005.

[18] S. Robertson and S. Walker. Some simple approximations to the 2-poisson model for probabilistic weighted retrieval. In *Proc. of Ann. Intl. ACM SIGIR Conf. on Research and Development in Information Retrieval*, 1994.

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

[20] M. Shokouhi, J. Zobel, S. Tahaghoghi, and F. Scholer. Using query logs to establish vocabularies in distributed information retrieval. *Information Processing and Management*, 43(1):169–180, 2007.

[21] 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.

[22] Z. Zheng, H. Zha, , K. Chen, and G. Sun. A regression framework for learning ranking functions using relative judgments. In *Ann. Intl. ACM SIGIR Conf. on Research and Development in Information Retrieval*, pages 287–294, 2007.

[23] Z. Zheng, H. Zha, T. Zhang, O. Chapelle, K. Chen, and G. Sun. A general boosting method and its application to learning ranking functions for web search. In *Proc. 21st Proc. of Advances in Neural Information Processing Systems*, 2007.