Beyond Ranking: Optimizing Whole-Page Presentation

Yue Wang[1]*, Dawei Yin[2], Luo Jie[3]†, Pengyuan Wang[2], Makoto Yamada[2,4],
Yi Chang[2], Qiaozhu Mei[1,5]
[1]Department of EECS, University of Michigan, Ann Arbor, MI, USA
[2]Yahoo Labs, 701 First Avenue, Sunnyvale, CA, USA
[3]Snapchat, Inc., 64 Market St, Venice, CA, USA
[4]Bioinformatics Center, Institute for Chemical Research, Kyoto University, Uji, Kyoto, Japan
[5]School of Information, University of Michigan, Ann Arbor, MI, USA
[1,5]{raywang, qmei}@umich.edu
[2]{daweiy, pengyuan, yichang}@yahoo-inc.com
[3]luoj.roger@gmail.com [2,4]myamada@kuicr.kyoto-u.ac.jp

ABSTRACT

Modern search engines aggregate results from different *verticals*: webpages, news, images, video, shopping, knowledge cards, local maps, etc. Unlike "ten blue links", these search results are heterogeneous in nature and not even arranged in a list on the page. This revolution directly challenges the conventional "ranked list" formulation in ad hoc search. Therefore, finding proper *presentation* for a gallery of heterogeneous results is critical for modern search engines.

We propose a novel framework that learns the optimal *page presentation* to render heterogeneous results onto search result page (SERP). Page presentation is broadly defined as the strategy to present a set of items on SERP, much more expressive than a ranked list. It can specify item positions, image sizes, text fonts, and any other styles as long as variations are within business and design constraints. The learned presentation is content-aware, i.e. tailored to specific queries and returned results. Simulation experiments show that the framework automatically learns eye-catchy presentations for relevant results. Experiments on real data show that simple instantiations of the framework already outperform leading algorithm in federated search result presentation. It means the framework can *learn* its own result presentation strategy purely from data, without even knowing the "probability ranking principle".

Categories and Subject Descriptors

H.3.3 [**Information Systems**]: Information Search and Retrieval; I.5.1 [**Pattern Recognition**]: Model

*This work was done when the first author was on an internship at Yahoo! Labs.

†This work was done when the third author was working at Yahoo Labs.

WSDM'16, February 22–25, 2016, San Francisco, CA, USA.
© 2016 ACM. ISBN 978-1-4503-3716-8/16/02…$15.00
DOI: http://dx.doi.org/10.1145/2835776.2835824

1. INTRODUCTION

A decade ago, search engines returned "ten blue links". Result presentation was straightforward: ranking webpages by estimated relevance. It naturally saves user effort as she scans down the list, hopefully hitting the desired information at top ranks. This "probability ranking principle" was long envisioned in the 1970s [36], and later confirmed by eye-tracking studies [20, 19] and search log analysis [24, 15].

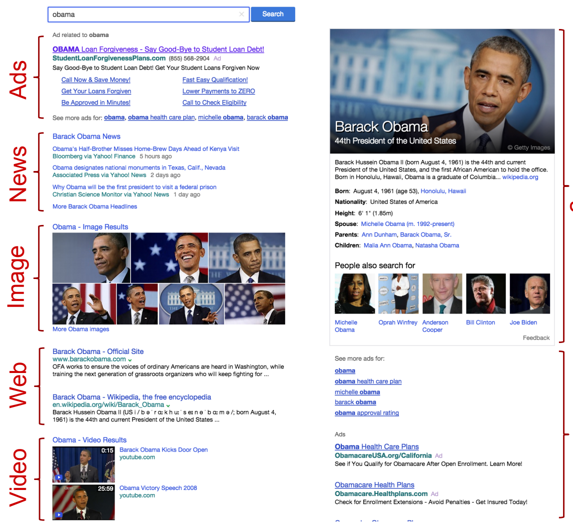
Today's search engines return far richer results than "ten blue links". Aside from webpages, results can also include news, images, video, shopping, structured knowledge, and local business maps. Each specific corpus is indexed by a *vertical* search engine; they are *federated* to serve the user's information need. Unlike "ten blue links", vertical search results have different visual appearance, layouts and sizes. They span across multiple columns on the page, not restricted in the mainline list (Figure 1).

Federated search results have been transforming user interaction patterns on search result pages (SERPs). Human eyeballs are spontaneously attracted by graphical results, causing a significant attention bias known as the *vertical bias* [12, 26, 31]. More interestingly, blue links surrounding a vertical result are also examined with increased probability [12]. In the presence of vertical results, user satisfaction towards an entire SERP cannot be reliably inferred from preference judgments for pairs of results [6].

These observations indicate that users do *not* sequentially scan results returned by federated search. Although the conventional "ranked list" formulation can still be used for federated search result presentation [5, 4], it is essentially a first-order approximation of the problem.

In this paper, we propose a novel framework that learns the optimal presentation for heterogeneous search results on SERP. Page presentation is broadly defined as the strategy to present a set of heterogeneous items on SERP, much more expressive than a ranked list. It can specify item positions, image sizes, text fonts, or any other styles as long as changes of these elements are allowed by business constraints [1] and page design templates. The goodness of a presentation is measured by user satisfaction metric: better presentation will make the user happier. The framework first learns a scoring function that maps search results and their presen-

[1]For example, sponsored results must be placed on the top.



Ad related to **obama**

OBAMA Loan Forgiveness - Say Good-Bye to Student Loan Debt!
StudentLoanForgivenessPlans.com (855) 568-2904 Ad
Say Good-Bye to Student Loan Debt! Get Your Student Loans Forgiven Now
Call Now & Save Money! Fast Easy Qualification!
Get Your Loans Forgiven Lower Payments to ZERO
Be Approved in Minutes! Call to Check Eligibility
See more ads for: **obama**, **obama** health care plan, michelle **obama**, barack **obama**

Barack Obama News
Obama's Half-Brother Misses Home-Brew Days Ahead of Kenya Visit
Bloomberg via Yahoo! Finance 5 hours ago
Obama designates national monuments in Texas, Calif., Nevada
Associated Press via Yahoo! News 2 days ago
Why Obama will be the first president to visit a federal prison
Christian Science Monitor via Yahoo! News 1 day ago
More Barack Obama Headlines

Obama - Image Results
More Obama images

Barack Obama - Official Site
www.barackobama.com
OFA works to ensure the voices of ordinary Americans are heard in Washington, while training the next generation of grassroots organizers who will keep fighting for ...

Barack Obama - Wikipedia, the free encyclopedia
en.wikipedia.org/wiki/Barack_Obama
Barack Hussein Obama II (US i / b ə ˈ r ɑː k h uː ˈ s eɪ n ə ˈ b ɑː m ə /; born August 4, 1961) is the 44th and current President of the United States ...

Obama - Video Results
0:15 Barack Obama Kicks Door Open
youtube.com
25:59 Obama Victory Speech 2008
youtube.com

Barack Obama
44th President of the United States
© Getty Images
Barack Hussein Obama II (born August 4, 1961) is the 44th and current President of the United States, and the first African American to hold the office. Born in Honolulu, Hawaii, Obama is a graduate of Columbia... wikipedia.org
Born: August 4, 1961 (age 53), Honolulu, Hawaii
Nationality: United States of America
Height: 6' 1" (1.85m)
Spouse: Michelle Obama (m. 1992-present)
Parents: Ann Dunham, Barack Obama, Sr.
Children: Malia Ann Obama, Natasha Obama
People also search for
Michelle Obama Oprah Winfrey Anderson Cooper Bill Clinton Joe Biden
Feedback

See more ads for:
obama
obama health care plan
michelle **obama**
barack **obama**
obama approval rating

Ads
Obama Health Care Plans
ObamacareUSA.org/California Ad
See if You Qualify for Obamacare After Open Enrollment. Learn More!

Obamacare Health Plans
Obamacare.Healthplans.com Ad
Check for Enrollment Extensions - Avoid Penalties - Get Insured Today!

Figure 1: Modern search engine result page.

tation on SERP to user satisfaction metric. Then, given search results of a new query, the framework computes a presentation that maximizes user satisfaction.

The framework is quite general. First, practitioners can flexibly define the scope of page presentation. It can encode item positions (both horizontal and vertical) as well as element styles, such as image sizes and text fonts. It naturally encompasses ranked list as a special case. Second, different application scenarios can adopt different user satisfaction metric. It is not limited to click-based metric, but can also take other interactive behaviors into account, such as dwelling time and time-to-first-click. Lastly, the framework can potentially be instantiated in other interactive search scenarios, such as presenting search results on mobile and tablet devices, displaying multimedia feeds in online social networks, and arranging items on retailing websites.

We conduct experiments on both synthetic and real data to demonstrate the potential power of the proposed framework. Simulation experiments show that the framework can adapt to different types of attention bias and learn to present relevant results to catch user's eyeballs. This means our approach directly targets the new challenge brought by federated search, where users may not scan the results sequentially, and results are not in a ranked list. In real data experiments, simple instantiations of the framework outperform the leading algorithm in federated search result ranking. This is encouraging, because ranking algorithms use the probability ranking principle in its result presentation, while our framework does not even know the existence of it. Nevertheless, it *learns* its own result presentation principle purely from data and is able to deliver the state-of-the-art performance.

Our main contribution is summarized as follows:

- We formulate a new problem, *whole-page presentation optimization*, which extends the homogeneous document ranking in ad hoc search;
- We propose a general framework that computes the optimal presentation for federated search results.
- Experiments on synthetic and real data demonstrate that the proposed framework is promising in solving the new problem.

2. PROBLEM FORMULATION

The problem statement of page presentation optimization is as follows: "given search results to be displayed on a page, to find the optimal presentation that maximizes user satisfaction". We assume the following setting: search engine returns a set of results upon receiving a query, and renders the items on SERP according to some presentation strategy. As the SERP is shown to the user, she interacts with it and get certain degree of satisfaction. Now let us define some important concepts in our setting:

Definition 1 (Page Content): Page content is the set of search results to be displayed on a page. Each search result is an **item**. Upon receiving user's query, search engine backend returns a set of k items. Each item is represented as a vector $\mathbf{x}_i$ [2]. Note that different users and different queries will produce different sets of items, so $\mathbf{x}_i$ also encodes information from actual users and queries. Page content is represented as concatenation of k item vectors: $\mathbf{x}^\top = (\mathbf{x}_1^\top, \cdots, \mathbf{x}_i^\top, \cdots, \mathbf{x}_k^\top)$. The domain of $\mathbf{x}$ is defined by all possible page content returned by the backend, denoted as $\mathcal{X}$.

Definition 2 (Page Presentation): Page presentation defines the way in which page content $\mathbf{x}$ is displayed, such as position, vertical type, size, and color. It is encoded as a vector $\mathbf{p}$. The domain of $\mathbf{p}$ is defined by all possible page presentations permitted by business and design constraints, denoted as $\mathcal{P}$.

Definition 3 (Search Result Page, SERP): When page content $\mathbf{x}$ is put on page according to presentation strategy $\mathbf{p}$, a search result page (SERP) is generated. In other words, content $\mathbf{x}$ and presentation $\mathbf{p}$ uniquely determine a SERP. It is represented as a tuple $(\mathbf{x}, \mathbf{p}) \in \mathcal{X} \times \mathcal{P}$.

Definition 4 (User Response): User response includes her actions on SERP, such as number of clicks, positions of clicks, dwell time of clicks, and time to first click. This information is encoded in a vector $\mathbf{y}$. The domain of $\mathbf{y}$ is defined by all possible user responses, denoted as $\mathcal{Y}$.

Definition 5 (User Satisfaction): The user experiences certain degree of satisfaction as she interacts with the SERP. We assume that user satisfaction can be calibrated as a real value $s \in \mathbb{R}$: larger value of s means higher satisfaction.

The user response is a strong indicator for satisfaction. Intuitively, if a user opened the SERP, clicked on the top result right away, then spent long time dwelling on that result, she was highly likely to be happy with the result. With definitions above, we formulate our problem:

Page Presentation Optimization is to find the presentation $\mathbf{p} \in \mathcal{P}$ for a given page content $\mathbf{x} \in \mathcal{X}$, such that when the SERP $(\mathbf{x}, \mathbf{p})$ is presented to the user, her satisfaction score s is maximized.

[2]Throughout the paper we use bold lowercase letters for *column* vectors.

If we assume that there exists a scoring function $F : \mathcal{X} \times \mathcal{P} \to \mathbb{R}$ that maps SERP $(\mathbf{x}, \mathbf{p})$ to user satisfaction score s, then page presentation optimization problem can be formally written as

$$\max_{\mathbf{p} \in \mathcal{P}} F(\mathbf{x}, \mathbf{p}),$$
$$\text{subject to constraints on presentation } \mathbf{p}.$$

The problem of page presentation optimization is both new and challenging. It is new because page presentation can be flexibly defined, which opens up possibility to learn brand-new ways to display information. Retrieval and recommender systems typically use a ranked list for displaying homogeneous content. As heterogeneous results are weaved onto webpages, it is critical to present them in a proper manner to maximize user's utility. The problem is challenging because it is rather unclear how to find the scoring function that maps the *entire* SERP (content and presentation) to user satisfaction. We propose our solution framework in the next section.

3. PRESENTATION OPTIMIZATION FRAMEWORK

We propose to solve page presentation optimization using a supervised learning approach. This section sets up a general framework for our approach, including data collection methodology, design of scoring function $F(\cdot, \cdot)$, the learning and optimization stages. In the next section, we describe actual instantiations of the framework.

3.1 Data Collection Through Exploration

Supervised machine learning needs labelled training data. The caveat in data collection here is that normal search traffic cannot be used as the training data to learn the scoring function $F(\mathbf{x}, \mathbf{p})$. This is because in normal search traffic, search engine has a deterministic policy to present page content $\mathbf{x}$, which is controlled by existing model/rules within the system. In other words, page presentation $\mathbf{p}$ is uniquely determined given page content $\mathbf{x}$. However, we expect the model F to tell us user satisfaction as we search through *different* page presentations. Confounding between $\mathbf{x}$ and $\mathbf{p}$ will bias the learned model, which will be a serious problem.

To eliminate confounding, we allocate "presentation exploration bucket" to do randomized experiments. For each request in the bucket, we organize page content with random page presentation. Here "random" means to uniformly draw presentation strategies within business and design constraints, such that user experience is not hurt too much. Further, the presentation exploration traffic is controlled within a very small amount so as not to affect overall quality of the search service. Data collected in this way allow unbiased estimation of the scoring function.

In cases that showing random exploration results to the users is not desired, it would also be possible to either hire human annotators to label the page, or collect data from multiple buckets with different fixed presentation strategy as every Internet company is doing for testing their UI changes. Since we have already developed a good data collection through exploration framework in our production system, we choose to take this approach for data collection.

3.2 Learning Stage

The core of page presentation optimization is to estimate the scoring function $s = F(\mathbf{x}, \mathbf{p})$. We might consider two approaches:

(I) ***Direct approach:*** Collect page-wise user satisfaction ratings and directly model the dependency between SERP and user satisfaction. The dependency path is "$(\mathbf{x}, \mathbf{p}) - s$".

(II) ***Factorized approach:*** First predict user response $\mathbf{y}$ on SERP, then find a function that measure user satisfaction from these responses. The dependency path is "$(\mathbf{x}, \mathbf{p}) - \mathbf{y} - s$".

Approach (I) is straightforward. However it is very difficult, particularly at a large scale, to obtain explicit user rating s towards the *entire* SERP. To construct such data set, we would have needed substantial amount of observations and human annotation to overcome training data sparseness.

Approach (II) takes two steps. The first step is to predict user responses on a given page; the second step is to measure user satisfaction based on her page-wise response. Introducing user response variable $\mathbf{y}$ permits a separation of concerns. On the one hand, user response on a page is a direct consequence of interacting with the page. On the other hand, user satisfaction is typically estimated from user responses *only*, e.g. using total number of clicks, or long dwell time. In Approach (II), the complex dependency in $F(\cdot, \cdot)$ is decomposed into two relatively independent factors. Furthermore, on a practical note, Approach (II) is more realistic for current Web technology because user response on SERP can be easily collected via Javascript, whereas explicitly asking the users to evaluate the whole page is very uncommon. Therefore, we adopt the ***factorized approach***.

In factorized approach, the first step is to learn a *user response model*

$$\mathbf{y} = f(\mathbf{x}, \mathbf{p}), \ \forall \ \mathbf{x} \in \mathcal{X}, \mathbf{p} \in \mathcal{P}.$$

This is a supervised learning task; the actual form of $f(\mathbf{x}, \mathbf{p})$ can be chosen flexibly. We can simply build one model for each component y_i in $\mathbf{y}$, or we can jointly predict all components of $\mathbf{y}$ using structured output prediction [10]. In any case, user's responses on the page depends on both the content (whether it is relevant, diverse, or attractive) and the presentation (whether it is close to the top, around a graphical block, or shown in big size).

The second step is a utility funciton which defines a *user satisfaction metric*

$$s = g(\mathbf{y}), \forall \ \mathbf{y} \in \mathcal{Y}.$$

Finding the right user satisfaction metric based on page-wise user responses is not the focus of this paper, and can itself be a substantial research topic in interactive information systems [21, 30, 38]. Indeed, practitioners often heuristically define the metric as aggregation of fine-grained user responses, such as click-through rates, long-dwell-time clicks, time-to-first-click.

Finally, our scoring function for the entire SERP is

$$s = F(\mathbf{x}, \mathbf{p}) = (g \circ f)(\mathbf{x}, \mathbf{p}) = g(f(\mathbf{x}, \mathbf{p})).$$

3.3 Optimization Stage

We compute the optimal presentation $\mathbf{p}^*$ given content $\mathbf{x}$ by solving the following optimization problem:

$$\max_{\mathbf{p}\in\mathcal{P}} g(f(\mathbf{x},\mathbf{p})),$$
$$\text{subject to constraints on presentation } \mathbf{p}.$$

Computational cost of this optimization problem depends on actual form of the objective function $F = g \circ f$ and the constraints on presentation $\mathbf{p}$. In the next section we show that for certain instantiations of f and g, $\mathbf{p}^*$ can be computed quite efficiently.

4. INSTANTIATIONS OF PRESENTATION OPTIMIZATION FRAMEWORK

This section describes instantiations of the framework, including feature representation, user satisfaction metric, two user response models and their learning and optimization stages. We conclude this section by showing that the framework encompasses learning to rank.

4.1 Features

Both content and presentation on a SERP are represented in a feature vector, which will be the input to user response models.

Content Features

Content features contain information of the query and corresponding search results, similar to those used in learning to rank. We adopt the same content features as used in [23] to facilitate direct comparison in experiments (Section 6):

- **Global result set features**: features derived from all returned results. They indicate the content availability of each vertical.
- **Query features**: lexical features such as the query unigrams, bigrams and co-occurrence statistics. We also use outputs of query classifiers, and historical session based query features, etc.
- **Corpus level features**: query-independent features derived for each vertical and web document such as historical click-through rates, user preferences, etc.
- **Search result features**: extracted from each search result. A list of statistical summary features such as relevance scores and ranking features of individual results. For some verticals, we also extract some domain specific meta features, such as if the movie is on-screen and if the movie poster is available in the movie vertical, and the number of hits for the news articles from the news vertical in the last few hours.

Presentation Features

Presentation features encode the way in which search results are displayed on SERP, which are novel features in our framework. Concrete examples include:

- **Binary indicators**: whether to show an item on a position. The scheme can encode positions in a wireframe, such as a list or multi-column panels. Let there be k positions in the frame, and k items to be displayed. Let i be the index of items, j be the index of positions, $1 \le i, j \le k$. The presentation of item i, $\mathbf{p}_i$, is a 1-of-k binary encoding vector. If document i is placed at position j, then the j-th component of $\mathbf{p}_i$ is 1 and all others are 0. In this case we denote the value of $\mathbf{p}_i$ as $p_{ij} = 1$. The page presentation $\mathbf{p}^\top = (\mathbf{p}_1^\top, \cdots, \mathbf{p}_k^\top)$ consists of $k \times k$ binary indicator variables, essentially encoding the permutation of k objects.
- **Categorical features**: discrete properties of page items, e.g., multimedia type of an item (shown as text or image), typeface of a textual item;
- **Numerical features**: continuous properties of page items, e.g. brightness and contrast of a graphical item.
- **Other features**: e.g. certain interactions between page content and presentation may affect user response, such as "a textual item immediately above a graphical item".

We use two types of presentation features in real data experiments. We encode positions of items with binary indicators. For the local search results, we encode presentation size as a categorical feature ("single" vs. "multiple" entries).

4.2 User Satisfaction Metric

We assume that user satisfaction metric $g(\mathbf{y})$ is in the form of weighted sum of components in $\mathbf{y}$:

$$g(\mathbf{y}) = \mathbf{c}^\top \mathbf{y}.$$

In experiments, we use the click-skip metric for k items [23]:

$$g(\mathbf{y}) = \sum_{i=1}^{k} y_i,$$

where $y_i = 1$ if item i is clicked, and $y_i = -1$ if item i is skipped *and* some item below is clicked. A skip often indicates wasted inspection, so we set it to be a unit of negative utility. This metric strongly prefers adjacent clicks at top positions.

4.3 User Response Models

We use two models for predicting page-wise user response. The first model takes as features quadratic interaction between content and presentation. It permits an efficient optimization stage. The second model uses gradient boosted decision trees to capture more complex, nonlinear interaction between content and presentation. We expect it to generate improved performance.

Quadratic Feature Model

First, let us consider a simple instantiation of user response model that has efficient solution in the optimization stage. Since it uses quadratic interaction features between $\mathbf{x}$ and $\mathbf{p}$, we call it *Quadratic Feature Model*.

Assume there are k positions for k items. Page content $\mathbf{x}$ is the concatenation of k item vectors; page presentation is encoded using binary indicators, $\mathbf{p} \in \{0,1\}^{k\times k}$, as defined in Section 4.1. The model also contains fully interaction between $\mathbf{x}$ and $\mathbf{p}$ as features. Let $\text{vec}(\mathbf{A})$ denote the row vector containing all elements in matrix $\mathbf{A}$, taken column by column, left to right. The augmented feature vector $\boldsymbol{\phi}$ for Quadratic Feature Model is:

$$\boldsymbol{\phi}^\top = (\mathbf{x}^\top, \mathbf{p}^\top, \text{vec}(\mathbf{x}\mathbf{p}^\top)).$$

Let $\mathbf{y} \in \mathbb{R}^k$ be the user response vector; each component y_i is a user response (e.g. click or skip) on item i. A linear model f_i is used to predict each y_i in $\mathbf{y}$:

$$y_i = f_i(\boldsymbol{\phi}) = \mathbf{w}_i^\top \boldsymbol{\phi} = \mathbf{u}_i^\top \mathbf{x} + \mathbf{v}_i^\top \mathbf{p} + \mathbf{x}^\top \mathbf{Q}_i \mathbf{p}. \tag{1}$$

$\mathbf{u}_i$, $\mathbf{v}_i$, and $\mathbf{Q}_i$ are coefficients for content-only, presentation-only, and content-presentation quadratic interaction features, respectively. The coefficients $\mathbf{w}_i = \{\mathbf{u}_i, \mathbf{v}_i, \mathbf{Q}_i\}$ can be estimated using regularized linear regression. To avoid overfitting, we regularize the L_2 norm of $\mathbf{u}_i$ and $\mathbf{v}_i$, and further impose low-rank regularization on $\mathbf{Q}_i$ to handle the sparsity issue of quadratic features.

In total, we will have k such models, each predicting one y_i in $\mathbf{y}$. To group the k models in notation, let us write coefficients as $\mathbf{U} = (\mathbf{u}_1, \cdots, \mathbf{u}_k)^\top, \mathbf{V} = (\mathbf{v}_1, \cdots, \mathbf{v}_k)^\top, \mathbf{Q} = \text{diag}(\mathbf{Q}_1, \cdots, \mathbf{Q}_k)$, and "copy" $\mathbf{x}$ and $\mathbf{p}$ k times to get the matrix $\mathbf{X} = \text{diag}(\mathbf{x}^\top, \cdots, \mathbf{x}^\top)$ and the vector $\mathbf{t}^\top = (\mathbf{p}^\top, \cdots, \mathbf{p}^\top)$. To clarify dimensionality, if $\mathbf{x} \in \mathbb{R}^n$, $\mathbf{p} \in \mathbb{R}^m$, then $\mathbf{U} \in \mathbb{R}^{k\times n}$, $\mathbf{V} \in \mathbb{R}^{k\times m}$, $\mathbf{X} \in \mathbb{R}^{k\times nk}$, $\mathbf{Q} \in \mathbb{R}^{nk\times mk}$, and $\mathbf{t} \in \mathbb{R}^{mk}$. The user response model can be written as

$$\mathbf{y} = f(\mathbf{x}, \mathbf{p}) = \mathbf{U}\mathbf{x} + \mathbf{V}\mathbf{p} + \mathbf{X}\mathbf{Q}\mathbf{t}.$$

Denote user satisfaction metric as $g(\mathbf{y}) = \mathbf{c}^\top \mathbf{y}$. Then the scoring function $F = g \circ f$ is

$$\begin{aligned} F(\mathbf{x}, \mathbf{p}) &= g(f(\mathbf{x}, \mathbf{p})) \\ &= \mathbf{c}^\top \mathbf{U}\mathbf{x} + \mathbf{c}^\top \mathbf{V}\mathbf{p} + \mathbf{c}^\top \mathbf{X}\mathbf{Q}\mathbf{t} \\ &= \mathbf{c}^\top \mathbf{U}\mathbf{x} + \mathbf{a}^\top \mathbf{p} \end{aligned} \tag{2}$$

where $\mathbf{a} = \mathbf{V}^\top \mathbf{c} + \sum_{i=1}^k c_i \mathbf{Q}_i^\top \mathbf{x}$ is a known vector.

To this end, the optimization stage is to find the $\mathbf{p}$ that maximizes (2) subject to the constraints on $\mathbf{p}$. Since page content $\mathbf{x}$ is given, the first term in (2) is a constant and can be dropped. The second term $\mathbf{a}^\top \mathbf{p}$ is a linear term of $\mathbf{p}$. Since $\mathbf{p} \in \{0,1\}^{k\times k}$ encodes a k-permutation, Each component in $\mathbf{a} \in \mathbb{R}^{k\times k}$ represents the gain of user satisfaction if item i is placed in position j, $1 \le i, j \le k$. Therefore, the optimization problem reduces to *maximum bipartite matching*, a special case of linear assignment problem. It can be efficiently solved by Hungarian algorithm [25] with time complexity $O(|\mathbf{p}|^3) = O(k^6)$. On a single-core computer with 2GHz CPU, the problem can be solved within 10 milliseconds for $k = 50$ items.

Gradient Boosted Decision Tree Model

In order to capture more complex, nonlinear interaction between content $\mathbf{x}$ and presentation $\mathbf{p}$, we replace the quadratic feature model f_i in previous section with a gradient boosted decision trees model h_i^{GBDT}. Gradient boosted decision trees (GBDT) is a very effective method for learning nonlinear functions [18].

Our feature vector is

$$\boldsymbol{\phi}^\top = (\mathbf{x}^\top, \mathbf{p}^\top),$$

and each user response y_i in $\mathbf{y}$ is predicted by a GBDT model:

$$y_i = h_i^{\text{GBDT}}(\mathbf{x}, \mathbf{p}).$$

The user satisfaction metric is $g(\mathbf{y}) = \mathbf{c}^\top \mathbf{y} = \sum_{i=1}^k c_i y_i$.

In optimization stage, since each h_i is now a nonparametric model, we cannot get the analytical form of $F(\mathbf{x}, \mathbf{p}) = \sum_{i=1}^k c_i h_i^{\text{GBDT}}(\mathbf{x}, \mathbf{p})$ in terms of $\mathbf{p}$. That is, the optimization over $\mathbf{p}$ is intractable. Nevertheless, in realistic settings, the search space of $\mathbf{p}$ is usually pruned down to tens of possible values by business and design constraints. We implement parallel enumeration to quickly find the optimal presentation that maximizes user satisfaction.

4.4 Special Case: Learning to Rank

When we restrict page presentation to be a ranked list, and assume that users are more satisfied if more relevant results are placed at top ranks, then presentation optimization reduces to the traditional ranking problem. We point out this connection to demonstrate the generality of the proposed framework.

The instantiation is as follows. We use binary indicators in Section 4.1 to represent the ranked list. Let user response $\mathbf{y}$ decompose into k components, each representing the user's utility of seeing result i at rank j_i. Let user response model $f(\mathbf{x}, \mathbf{p})$ decompose into k real-valued component, each only taking as input $\mathbf{x}_i$ and its rank j_i. So we have

$$\begin{aligned} f(\mathbf{x}, \mathbf{p}) &= f(\mathbf{x}_1, \cdots, \mathbf{x}_k, \mathbf{p}_1, \cdots, \mathbf{p}_k) \\ &= (f_1(\mathbf{x}_1, \mathbf{p}_1), \cdots, f_k(\mathbf{x}_k, \mathbf{p}_k))^\top \\ &= (f_1(\mathbf{x}_1, p_{1j_1} = 1), \cdots, f_k(\mathbf{x}_k, p_{kj_k} = 1))^\top. \end{aligned} \tag{3}$$

Typically, the ranking function $h(\mathbf{x}_i)$ of result i is position-independent. It can also be interpreted as the score of result i seen on the top rank ($j_i = 1$). That means

$$h(\mathbf{x}_i) = f_i(\mathbf{x}_i, p_{i1} = 1).$$

Furthermore, traditional ranking problem assumes that the utility of a result is discounted by a factor w_j if it is ranked at position j. $w_j > 0$ is a decreasing function of j. E.g. in discounted cumulative gain (DCG),

$$w_j = \frac{1}{\log_2(1+j)}.$$

The discounting assumption implies:

$$\begin{aligned} f_i(\mathbf{x}_i, p_{ij} = 1) &= w_j f_i(\mathbf{x}_i, p_{i1} = 1) \\ &= w_j h(\mathbf{x}_i). \end{aligned} \tag{4}$$

Combining (3) and (4), user response model is realized as

$$f(\mathbf{x}, \mathbf{p}) = (w_{j_1} h(\mathbf{x}_1), \cdots, w_{j_k} h(\mathbf{x}_k))^\top,$$

where $h(\cdot)$ is the ranking function. User satisfaction is measured by the quality of ranked list, which *accumulate* the gain at each position:

$$g(\mathbf{y}) = \sum_{i=1}^k y_i = \sum_{i=1}^k w_{j_i} h(\mathbf{x}_i).$$

Clearly, maximum user satisfaction $g(\mathbf{y})$ is always achieved by *sorting the results by descending relevance scores.* This is the default presentation strategy of learning to rank.

5. SIMULATION STUDY

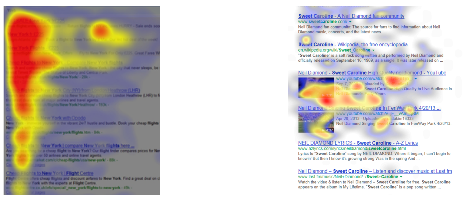
We demonstrate potential capability of presentation optimization framework by simulation. We use synthetic dataset so that we know the "ground truth" mechanism to maximize user satisfaction, and we can easily check whether the algorithm can indeed learn the optimal page presentation to maximize user satisfaction. We have two goals in this study:

(1) We show that the framework enables general definition of page presentation.

(2) We use both position bias and item-specific bias to show that the framework can automatically adapt to user interaction habits.

5.1 Overview

We first give a brief overview of simulation workflow. The simulated "presentation exploration bucket" will generate a page containing a set items with random presentation. Every time a new page is generated, each item is assigned some amount of reward (e.g. relevant information) drawn from an underlying distribution. The simulated "user" will have a certain type of attention bias: (1) position bias, in which more attention is paid to certain region of the page than elsewhere (Figure 2a); (2) vertical bias, or item-specific bias, in which more attention is attracted by a specific type of item and its surrounding area (Figure 2b).



(a) Position bias [2]   (b) Vertical bias [1]

Figure 2: Different types of user attention bias.

An "interaction" happens when the "presentation exploration bucket" generates a page and the "user" examines it with attention bias. When the user examines an item, she receives the corresponding reward. User satisfaction towards the page is the sum of rewards. The page content, presentation, as well as the examined items and positions (user responses), become data that the framework learns from. Finally, we test if the framework successfully learned user's attention bias. Given a new set of items, we expect to see that the framework will place items with higher rewards to positions with more concentrated attention to achieve maximum user satisfaction. Therefore, to visualize the model's current belief in user attention bias, we can plot the distribution of item rewards on the page.

5.2 Data Generating Process

On the "search engine" side, a page (either 1-D list or 2-D grid) contains k positions. The page content $\mathbf{x} = (x_1, \cdots, x_k)^\top$, $x_i \sim \mathcal{N}(\mu_i, \sigma)$ represents intrinsic rewards of k items. We set $k = 10$ for 1-D list and $k = 7 \times 7$ for 2-D grid. μ_i's are random numbers drawn from in $[0, 1]$, $\sigma = 0.1$. The page presentation $\mathbf{p}$ is drawn from length-k permutations uniformly at random. The whole page is represented as $(\mathbf{x}, \mathbf{p})$.

On the "user" side, attention bias is simulated as follows:

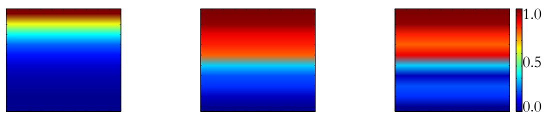
Position bias: whether to examine position j is a Bernoulli random variable with parameter p_j. A real-life example is the top-down position bias, commonly observed when the user interacts with a ranked list.

Item-specific bias: whether to examine item i is a Bernoulli random variable with parameter p_i. A real-life example is the vertical bias, commonly observed when the user interacts with a page that contains vertical search results (images, videos, maps, etc).

Then, the "user" will "interact" with the page $(\mathbf{x}, \mathbf{p})$: k binary values are drawn from k Bernoulli distributions, and recorded as a user response vector $\mathbf{y} \in \{0, 1\}^k$. if item i is examined, $y_i = 1$, the user receives a reward x_i. The user satisfaction equals the sum of reward of examined items. We generate 100,000 pages to train the Quadratic Feature Model described in Section 4.3.

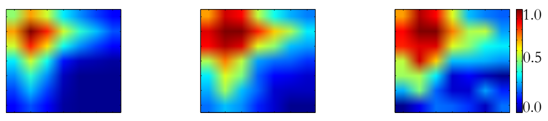
5.3 Discussion

To visualize the learned optimal presentation, we pick a random $\mathbf{x}$ and compute the corresponding optimal presentation $\mathbf{p}^*$, then arrange the x_i's according to $\mathbf{p}^*$. A page is visualized as a heat map of x_i's rewards. Ideally, the items with higher reward ("better content") should be placed onto the position with higher probability of user attention.



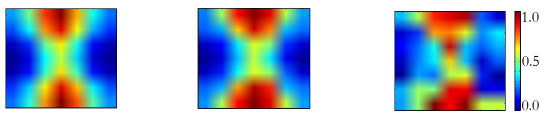
(a) Position bias   (b) Ideal presentation   (c) Learned presentation

Figure 3: Top position bias and presentation on 1-D list.



(a) Position bias   (b) Ideal presentation   (c) Learned presentation

Figure 4: Top-left position bias and presentation on 2-D canvas.

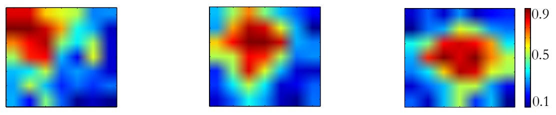


(a) Position bias   (b) Ideal presentation   (c) Learned presentation

Figure 5: Two-end position bias and presentation on 2-D canvas.

Figure 3, 4, and 5 visualize the presentation results under various position biases. We can see that the algorithm indeed learns to put "better content" to positions with more user attention. Because the definition of page presentation is general, it is able to handle both 1-D list and 2-D grid. Furthermore, it can capture complicated distribution of position bias on a 2-D canvas: the top-left position bias in Figure 4, and the top-bottom position bias in Figure 5.

Figure 6 visualizes the result under item-specific bias. This is an interesting case where an item on the page is very eye-catchy, and it also attracts user's attention to its surrounding items (e.g., an image attracts user's eyeballs on itself as well as its caption and description text). Also, suppose that for items farther away from that eye-catchy item, the user's



(a) $s = 4.07$ (b) $s = 4.92$ (c) $s = 5.41$

Figure 6: Item-specific bias. s: page-wise user satisfaction. When a specific item (e.g. image) attracts user attention to not only itself but also its surrounding results, then the page-wise reward is highest when the vertical is placed at the center of the page.

attention drops further down. Then the optimal presentation strategy is to place the item on the *center* of the page, so that the whole page delivers the most reward. In Figure 6, we see that user satisfaction value s is highest when the item (the dark red region) is centered on the page.

6. REAL DATA EXPERIMENTS

We demonstrate the effectiveness of page presentation optimization framework by conducting experiments on the real-world data set collected via a commercial search engine.

6.1 Data Collection

We use a very small fraction of search traffic as the presentation exploration buckets. The data was collected through the year 2013. Vertical search results whose presentation are explored include *news*, *shopping*, and *local listings*. In the exploration buckets, the order of Web results are kept untouched and verticals are randomly slotted into allowed positions with uniform probability. Randomly generated SERPs are not influenced by any ranking algorithm in the system. As pointed out in Section 3.1, this is required to eliminate page content confounding when training models. The exploration SERP is then presented to the user who interacts with it in a normal fashion. Users response on the SERP, along with page-wise content information like the query, document features from backends, are logged.

6.2 Methods

We use two pointwise ranking models as baseline method. They are trained using the content features as described in Section 4.1. The first baseline method has been adopted in production (LOGIT-RANK) [23]. It estimates a logistic regression model for each vertical result (including web result):

$$y_i = \sigma(\mathbf{w}_i^\top \mathbf{x}_i),$$

where y_i is a binary target variable that indicates whether the result is clicked ($y_i = +1$) or skipped ($y_i = -1$) as described in Section 4.2, and $\sigma(\cdot)$ is the sigmoid link function rescaled to $[-1, 1]$.

The second baseline method uses gradient boosted decision trees to learn a pointwise ranking function (GBDT-RANK). This is essentially replacing the logistic regressor in LOGIT-RANK with a GBDT regressor:

$$y_i = h_i^{\text{GBDT}}(\mathbf{x}_i).$$

We evaluate the two instantiations of presentation optimization framework described in Section 4.3: the Quadratic Feature Model (QUAD-PRES) and the GBDT Model (GBDT-PRES). They use page-wise information $(\mathbf{x}, \mathbf{p})$ to predict the user response *vector*, i.e. the vector of clicks and skips.

Table 1: Match rate between random exploration presentation p and predicted optimal presentation $\mathbf{p}^*$. "Until Web$_1$ " means that p and $\mathbf{p}^*$ encode the same presentation above the 1st webpage result.

	Until WEB$_1$	Until WEB$_2$	Until WEB$_3$
LOGIT-RANK	68.68%	46.76%	30.85%
QUAD-PRES	71.63%	50.68%	33.42%

In implementation, we use Vowpal Wabbit [27] to learn logistic regression models, and XGBoost [13] to learn the gradient boosted decision tree models. The hyperparameters of the models are tuned on a small holdout data set.

6.3 Evaluation

We use half of the exploration SERP as training set (January – June), the rest as test set. It contains hundreds of millions of pageviews and was collected from real search traffic. Compared to standard supervised learning setup, it is difficult to do an unbiased offline performance evaluation because of the interactive nature of the task (see Section 4.3 in [23]). This is because the offline data $(\mathbf{x}^{(n)}, \mathbf{p}^{(n)}, \mathbf{y}^{(n)})$ is collected using a particular logging policy, so we only observe user response $\mathbf{y}^{(n)}$ for a specific page presentation $\mathbf{p}^{(n)}$. In offline evaluation, when the algorithm is given page content $\mathbf{x}^{(n)}$, it may output a presentation $\mathbf{p}^{*(n)} \neq \mathbf{p}^{(n)}$, for which we do not observe user response, hence cannot evaluate its goodness. To address this problem, we use an offline policy evaluation method proposed by Li et al. [28] for evaluating online recommendation systems. It is simple to implement and provides an unbiased performance estimate, thanks to data collected through random exploration. Given a stream of events $(\mathbf{x}^{(n)}, \mathbf{p}^{(n)}, \Pr(\mathbf{p}^{(n)}), \mathbf{y}^{(n)})$ collected through random exploration, where $\Pr(\mathbf{p}^{(n)})$ is the probability for the SERP $(\mathbf{x}^{(n)}, \mathbf{p}^{(n)})$ to be generated from uniform random exploration, the average user satisfaction for N offline events can be computed as

$$\bar{s} = \frac{1}{N} \sum_{n=1}^{N} \frac{g(\mathbf{y}^{(n)}) \mathbf{1}_{\{\mathbf{p}^{*(n)} == \mathbf{p}^{(n)}\}}}{\Pr(\mathbf{p}^{(n)})},$$

where $\mathbf{1}_{\{\cdot\}}$ is the indicator function, and $g(\mathbf{y}^{(n)})$ is user satisfaction towards SERP n. This means the algorithm is evaluated on those exploration SERPs whose presentation *matches* what is chosen by the algorithm; otherwise the SERP is discarded in offline evaluation.

As the match goes deeper down the page, the match rate decreases (Table 1). If we require *exact match* between predicted $\mathbf{p}^{*(n)}$ and actual $\mathbf{p}^{(n)}$, a large fraction of test set will be discarded and the performance estimates tend to have large variance hence unreliable. Our evaluation only focuses on vertical results shown above the first, second, and third webpage result. Note that the first webpage result is *not* always on top rank; the top rank is frequently occupied by the vertical results.

6.4 Results

Table 2 shows the average page-wise user satisfaction. It is encouraging to see that whole-page optimization methods outperform ranking methods, because ranking methods utilize probability ranking principle to rank results by relevance, which assumes a top-down position bias. Quad-Pres and Gbdt-Pres do not make this assumption, but *learns* its own result presentation principle purely from data. The reason that GBDT models work better than logistic regression models, mainly because logistic regression assumes linear decision boundary, while GBDT is capable of modeling nonlinear decision boundary.

Table 2: Average user satisfaction ($\times 10^{-3}$).

	Until Web$_1$	Until Web$_2$	Until Web$_3$
Logit-Rank	-0.25	1.79	1.89
Gbdt-Rank	2.18	3.68	2.22
Quad-Pres	0.62	6.39	5.37
Gbdt-Pres	**2.68**	**6.72**	**8.24**

Note that in our definition of user satisfaction metric $g(\mathbf{y})$, a skip causes negative utility ($y_i = -1$). The fact that Quad-Pres and Gbdt-Pres generally work better than the baseline methods is because they take into consideration the retrieved items, the page presentation, and their interaction on the entire SERP, not just a single result. The presentation-blind models Logit-Rank and Gbdt-Rank always want to put on top the results that will most probably gets clicked. However, for certain queries people might intentionally skip the graphical results (e.g., when shopping ads are shown when the search intent is in fact informational). In such cases, a click tends to happen *below* the top rank. In contrast, presentation optimization methods will consider both the result and its position on the page. That leads to more sensible arrangement of results. We see that Gbdt-Pres attracts more clicks and has less skips when we evaluate deeper down the SERP.

Table 3, 4, and 5 shows the click-through rate (CTR) above Web$_1$, Web$_2$, and Web$_3$, respectively. "S. Local" means a single entry of local business result (such as restaurants); "M. Local" means multiple entries of local business results. They are the same vertical/genre results presented in different sizes. In terms of CTR, ranking methods have very strong performance because they are directly optimized for high CTR. However, whole-page optimization methods still achieve competitive or sometimes better CTR by taking into account page-wise information.

It is interesting to see that for News vertical, there is not much help to know about other results on the SERP, neither their presentation. In contrast, knowing page-wise results helps improve the CTR of top-ranked local listings by a large margin. A possible explanation is that news, more like general webpages, contain rich text information and their content relevance can be readily modeled by standard ranking functions. On the other hand, local listings are in drastically different nature compared to webpages and news, therefore knowing the complementary information from other webpage results helps predicting the click/skip patterns. We can also observe small improvements in CTR of the shopping results. Since the shopping results shown on the top are most likely to be skipped, the algorithm learns to become extremely conservative in showing shopping verticals on top. This leads to a much smaller coverage of shopping results in the entire traffic.

As the match goes deeper down the page, the local ranking algorithms show decreased performance in terms of CTR. This is because the local ranking methods tend to greedily put the high CTR items on top of the page, but ignores the content on the entire page. In contrast, the page presentation algorithms, especially Gbdt-Pres, still get good CTR on News and Multi-local verticals, which takes larger coverage. This is attributed to the fact that they model user response over the entire page.

Table 3: CTR, match until Web$_1$

	News	Shopping	S. Local	M. Local
Coverage	0.46%	0.02%	0.02%	0.71%
Logit-Rank	21.05%	40.79%	11.58%	30.02%
Gbdt-Rank	23.28%	53.32%	38.26%	52.27%
Quad-Pres	21.97%	49.85%	47.16%	39.93%
Gbdt-Pres	22.34%	46.15%	48.12%	49.18%

Table 4: CTR, match until Web$_2$

	News	Shopping	S. Local	M. Local
Coverage	2.0%	0.11%	0.03%	2.3%
Logit-Rank	16.44%	23.71%	18.51%	8.92%
Gbdt-Rank	16.31%	30.39%	36.73%	23.11%
Quad-Pres	14.78%	13.57%	23.39%	27.53%
Gbdt-Pres	16.21%	40.83%	33.18%	35.23%

Table 5: CTR, match until Web$_3$

	News	Shopping	S. Local	M. Local
Coverage	3.8%	0.18%	0.11%	3.4%
Logit-Rank	14.52%	21.48%	13.80%	9.65%
Gbdt-Rank	12.51%	42.96%	24.93%	22.42%
Quad-Pres	11.45%	12.88%	15.47%	24.38%
Gbdt-Pres	14.11%	36.00%	24.72%	30.66%

7. RELATED WORK

As a general framework for search result presentation, this work draws on various aspects in IR research. It is inspired by the task of federated search result presentation. It extends traditional "ranked list" formulation to more a general notion of "presentation". Finally, it delivers optimal presentation by learning from interactive search logs.

7.1 Document Ranking in Retrieval

Document ranking has long been the core problem of ad hoc retrieval. Given a query, the retrieval system returns a list of documents ranked by decreasing probability of relevance. The presentation is optimal with respect to the user's effort when she sequentially and independently examines results from top to bottom [36]. Design and evaluation of document ranking functions are at the central stage of IR research, dating from vector space models [37] and language modeling ranking functions [40] to more recent machine learning ranking [29] and top document reranking [11, 22, 35].

Our framework extends homogeneous document ranking to heterogeneous content presentation. Document ranking is a special case when presentation is a ranked list. From an algorithmic perspective, we use *all* documents on SERP to determine the optimal presentation, which is in the same spirit of reranking/global ranking [35, 22]. The difference is that our framework allows much more general notion of presentation than a list. In fact, global ranking algorithms, and more broadly, structured prediction algorithms in machine learning literature [10], can be readily plugged into our framework as the user response model.

7.2 Federated Search

Federated search (or aggregated search) refers to searching through a collection of specialized search engines, *verticals*, and aggregating the results on SERP. Usually, contents from different verticals are heterogeneous and visually rich. Federated search has two sub-tasks: vertical selection and result presentation [32, 3]. Given a query, the task of vertical selection is to accurately determine which candidate verticals provide potentially relevant results [7, 8]. After getting results from candidate verticals, the task of result presentation is to merge vertical results with general webpage results on the same page [5].

This paper is concerned with result presentation. Previous approaches formulate it as a ranking problem [5, 34, 23]. Specifically, [5, 23] employ pointwise ranking functions to rank results and blocks, while [5, 34] also construct pairwise preference judgments to train a ranking function. [14] considers 2-D grid presentation for image and shopping results. Our framework allows more flexible definition of presentation than ranked list and 2-D grid, e.g. arbitrary frames, image sizes, and text fonts.

Federated search results significantly change the landscape of SERP, which in turn calls for changes in evaluation methodology. Bailey et al. [9] propose the notion of *whole-page relevance*. They argue that the Cranfield-style evaluation is inadequate to quantify user's holistic experience on modern SERP, such as overall presentation and coherence. It proposes to evaluate whole-page relevance by assigning grades to various SERP elements. Our framework incorporates this idea by defining an appropriate user satisfaction metric that guides the search of optimal presentation.

Our work is related to the whole-page optimization for sponsored search [16] or online ads bidding [33], where the focus is to optimize revenue of the search service provider. Our framework is more general and can be applied in these problems by changing the optimization objective.

7.3 Search Behavior Modeling

To deliver good search experience, understanding user behavior on SERP is critical. Eye-tracking experiments [20, 19] and click log analyses [24, 15] observe that users follow sequential order in browsing blue-link-only SERPs. Lower-ranked results are examined with lower probability. On the one hand, these results confirm the probability ranking principle, encouraging search engines to put more relevant results on top. On the other hand, the position bias has to be handled with care when using click-through data as relevance judgments to train ranking functions [24].

As heterogeneous results appear on modern SERP, users do not necessarily follow sequential order in browsing search results. Studies observe more interesting user interaction patterns, notably vertical bias [12, 26, 31] and presentation bias [39]. These patterns not only make it questionable to stick to the probability ranking principle, but also complicates the inference of relevance judgments from click-through data. As a result, global ranking that takes into account relations between documents is developed [32], and click modeling of federated search becomes even more elaborate [12, 17].

Instead of building click models, our framework uses randomized experiments to understand user's presentation preference. Using presentation exploration buckets, we collect user responses on almost identical contents rendered with different presentations. Interaction logs collected in this way are almost free from presentational bias and can be used to estimate user response to a specific presentation given the search content[3]. This enables automatic generation of presentations for different contents, which is more scalable than hand-crafted presentation rules.

8. CONCLUSION

This paper proposes a new problem in web search: *whole-page presentation optimization*. Modern retrieval systems return a collection of heterogeneous results, calling for more flexible presentation strategies than the conventional ranked list. This paper formulates the problem as a mathematical optimization problem, and proposes a framework that solves the problem in a data-driven fashion. This general framework is instantiated to enable more flexible and expressive definition of page presentation than ranked list. Simple instantiations of the framework are shown to outperform ranking-based methods in satisfying federated search users.

This study opens up many interesting avenues for the future work. We can instantiate the general presentation optimization framework properly on other heterogeneous content presentation scenarios, such as mobile and tablet search, where user's attention bias may be different from that on large screens. User response models can be instantiated in more sophisticated ways, e.g., modeling cursor position as conditional random fields over the SERP canvas. Finally, defining a proper quantitative metric for user satisfaction based on SERP-level, fine-grained user behaviors can be explored in human computer interaction research.

9. ACKNOWLEDGEMENTS

We sincerely thank the anonymous reviewers for their useful comments. This work is partially supported by the National Science Foundation under grant number IIS-1054199.

10. REFERENCES

[1] Eye tracking study: Google results with videos. https://www.looktracker.com/blog/eye-tracking-case-study/google-results-with-videos-eye-tracking-study.

[2] Google ads in second position get more attention. http://miratech.com/blog/eye-tracking-google.html.

[3] Trec federated web search track. https://sites.google.com/site/trecfedweb.

[4] J. Arguello and F. Diaz. Vertical selection and aggregation. In B. Long and Y. Chang, editors, *Relevance Ranking for Vertical Search Engines*. Elsevier, 2013.

[5] J. Arguello, F. Diaz, and J. Callan. Learning to aggregate vertical results into web search results. In *Proceedings of the 20th ACM conference on Information and knowledge management*, pages 201–210, 2011.

[6] J. Arguello, F. Diaz, J. Callan, and B. Carterette. A methodology for evaluating aggregated search results. In

[3]The presentation can never be completely random due to constraints of business and SERP template itself.

Advances in information retrieval, pages 141–152. Springer, 2011.

[7] J. Arguello, F. Diaz, J. Callan, and J.-F. Crespo. Sources of evidence for vertical selection. In *Proceedings of the 32nd ACM SIGIR conference on Research and development in information retrieval*, pages 315–322, 2009.

[8] J. Arguello, F. Diaz, and J.-F. Paiement. Vertical selection in the presence of unlabeled verticals. In *Proceedings of the 33rd ACM SIGIR conference on Research and development in information retrieval*, pages 691–698, 2010.

[9] P. Bailey, N. Craswell, R. W. White, L. Chen, A. Satyanarayana, and S. M. Tahaghoghi. Evaluating whole-page relevance. In *Proceedings of the 33rd ACM SIGIR conference on Research and development in information retrieval*, pages 767–768, 2010.

[10] G. Bakir, T. Hofmann, B. Schölkopf, A. Smola, B. Taskar, and S. Vishwanathan. Predicting structured data. 2007.

[11] J. Carbonell and J. Goldstein. The use of mmr, diversity-based reranking for reordering documents and producing summaries. In *Proceedings of the 21st ACM SIGIR conference on Research and development in information retrieval*, pages 335–336, 1998.

[12] D. Chen, W. Chen, H. Wang, Z. Chen, and Q. Yang. Beyond ten blue links: enabling user click modeling in federated web search. In *Proceedings of the fifth ACM conference on Web search and data mining*, pages 463–472, 2012.

[13] T. Chen. extreme gradient boosting library. https://github.com/tqchen/xgboost.

[14] F. Chierichetti, R. Kumar, and P. Raghavan. Optimizing two-dimensional search results presentation. In *Proceedings of the fourth ACM conference on Web search and data mining*, pages 257–266, 2011.

[15] N. Craswell, O. Zoeter, M. Taylor, and B. Ramsey. An experimental comparison of click position-bias models. In *Proceedings of the 2008 Conference on Web search and data mining*, pages 87–94, 2008.

[16] N. R. Devanur, Z. Huang, N. Korula, V. S. Mirrokni, and Q. Yan. Whole-page optimization and submodular welfare maximization with online bidders. In *Proceedings of the fourteenth ACM conference on Electronic commerce*, pages 305–322, 2013.

[17] F. Diaz, R. W. White, G. Buscher, and D. Liebling. Robust models of mouse movement on dynamic web search results pages. In *Proceedings of the 22nd ACM conference on information and knowledge management*, pages 1451–1460, 2013.

[18] J. H. Friedman. Stochastic gradient boosting. *Computational Statistics & Data Analysis*, 38(4):367–378, 2002.

[19] H. Gord. Eye tracking on universal and personalized search. http://searchengineland.com/eye-tracking-on-universal-and-personalized-search-12233. Technical report, Enquiro Research.

[20] L. A. Granka, T. Joachims, and G. Gay. Eye-tracking analysis of user behavior in www search. In *Proceedings of the 27th ACM SIGIR conference on Research and development in information retrieval*, pages 478–479, 2004.

[21] M. Hearst. The evaluation of search user interfaces. In *Search User Interfaces*. Cambridge University Press, 2019.

[22] S. Ji, K. Zhou, C. Liao, Z. Zheng, G.-R. Xue, O. Chapelle, G. Sun, and H. Zha. Global ranking by exploiting user clicks. In *Proceedings of the 32nd ACM SIGIR conference on Research and development in information retrieval*, pages 35–42, 2009.

[23] L. Jie, S. Lamkhede, R. Sapra, E. Hsu, H. Song, and Y. Chang. A unified search federation system based on online user feedback. In *Proceedings of the 19th ACM SIGKDD conference on Knowledge discovery and data mining*, pages 1195–1203. ACM, 2013.

[24] T. Joachims, L. Granka, B. Pan, H. Hembrooke, and G. Gay. Accurately interpreting clickthrough data as implicit feedback. In *Proceedings of the 28th ACM SIGIR conference on Research and development in information retrieval*, pages 154–161, 2005.

[25] H. W. Kuhn. The hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, 2(1-2):83–97, 1955.

[26] D. Lagun and E. Agichtein. Effects of task and domain on searcher attention. In *Proceedings of the 37th ACM SIGIR conference on Research and development in information retrieval*, pages 1087–1090, 2014.

[27] J. Langford. Vowpal wabbit. http://hunch.net/ vw.

[28] L. Li, W. Chu, J. Langford, and R. E. Schapire. A contextual-bandit approach to personalized news article recommendation. In *Proceedings of the 19th conference on World Wide Web*, pages 661–670, 2010.

[29] T.-Y. Liu. Learning to rank for information retrieval. *Foundations and Trends in Information Retrieval*, 3(3):225–331, 2009.

[30] Y. Liu, Y. Chen, J. Tang, J. Sun, M. Zhang, S. Ma, and X. Zhu. Different users, different opinions: Predicting search satisfaction with mouse movement information. In *Proceedings of the 38th ACM SIGIR conference on Research and development in information retrieval*, 2015.

[31] Z. Liu, Y. Liu, K. Zhou, M. Zhang, and S. Ma. Influence of vertical result in web search examination. In *Proceedings of the 38th ACM SIGIR conference on Research and development in information retrieval*, 2015.

[32] B. Long and Y. Chang. *Relevance Ranking for Vertical Search Engines*. Elsevier, 2014.

[33] P. Metrikov, F. Diaz, S. Lahaie, and J. Rao. Whole page optimization: how page elements interact with the position auction. In *Proceedings of the fifteenth ACM conference on Economics and computation*, pages 583–600, 2014.

[34] 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 *Proceedings of the fourth ACM conference on Web search and data mining*, pages 715–724, 2011.

[35] T. Qin, T.-Y. Liu, X.-D. Zhang, D.-S. Wang, and H. Li. Global ranking using continuous conditional random fields. In *Advances in neural information processing systems*, pages 1281–1288, 2009.

[36] S. E. Robertson. The probability ranking principle in ir. In *Journal of Documentation*, pages 294–304, 1977.

[37] G. Salton and C. Buckley. Term-weighting approaches in automatic text retrieval. *Information processing & management*, 24(5):513–523, 1988.

[38] A. Schuth, K. Hofmann, and F. Radlinski. Predicting search satisfaction metrics with interleaved comparisons. In *Proceedings of the 38th ACM SIGIR conference on Research and development in information retrieval*, 2015.

[39] Y. Yue, R. Patel, and H. Roehrig. Beyond position bias: Examining result attractiveness as a source of presentation bias in clickthrough data. In *Proceedings of the 19th conference on World Wide Web*, pages 1011–1018, 2010.

[40] C. Zhai and J. Lafferty. A study of smoothing methods for language models applied to ad hoc information retrieval. In *Proceedings of the 24th ACM SIGIR conference on Research and development in information retrieval*, pages 334–342, 2001.