# Adaptation of Offline Vertical Selection Predictions in the Presence of User Feedback

Fernando Diaz
Yahoo! Labs Montreal
1000 Rue de la Gauchetiere
Suite 2400
Montreal, QC H3M4W5
diazf@yahoo-inc.com

Jaime Arguello*
Language Technologies Institute
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213, USA
jaime@cs.cmu.edu

## ABSTRACT

Web search results often integrate content from specialized corpora known as *verticals*. Given a query, one important aspect of aggregated search is the selection of relevant verticals from a set of candidate verticals. One drawback to previous approaches to vertical selection is that methods have not *explicitly* modeled user feedback. However, production search systems often record a variety of feedback information. In this paper, we present algorithms for vertical selection which adapt to user feedback. We evaluate algorithms using a novel simulator which models performance of a vertical selector situated in realistic query traffic.

## Categories and Subject Descriptors

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

## General Terms

Algorithms

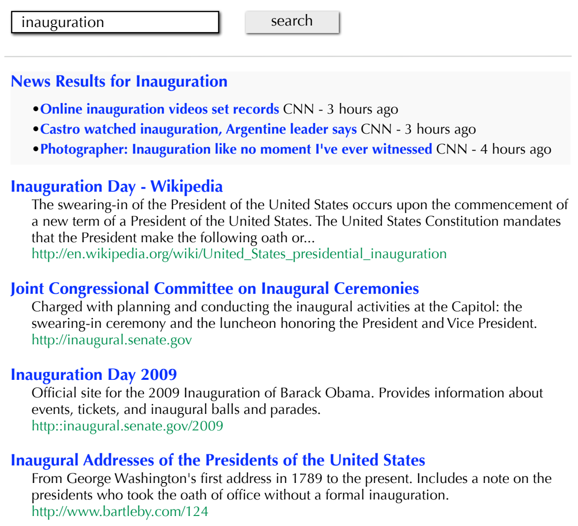
## Keywords

vertical selection, distributed information retrieval, resource selection, aggregated search, user feedback, simulation

## 1. INTRODUCTION

Traditional web search engines retrieve a ranked list of URLs in response to a user's query. Increasingly, search results pages include content from specialized sub-collections; this model has been referred to as *aggregated search* [18]. These sub-collections—referred to as *verticals*—include non-text media collections such as images and videos as well as genre-specific subsets of the web such as news and blogs. Given a query, relevant vertical content is often displayed

*Work conducted at Yahoo! Labs.



News Results for Inauguration
- Online inauguration videos set records CNN - 3 hours ago
- Castro watched inauguration, Argentine leader says CNN - 3 hours ago
- Photographer: Inauguration like no moment I've ever witnessed CNN - 4 hours ago

Inauguration Day - Wikipedia
The swearing-in of the President of the United States occurs upon the commencement of a new term of a President of the United States. The United States Constitution mandates that the President make the following oath or...
http://en.wikipedia.org/wiki/United_States_presidential_inauguration

Joint Congressional Committee on Inaugural Ceremonies
Charged with planning and conducting the inaugural activities at the Capitol: the swearing-in ceremony and the luncheon honoring the President and Vice President.
http://inaugural.senate.gov

Inauguration Day 2009
Official site for the 2009 Inauguration of Barack Obama. Provides information about events, tickets, and inaugural balls and parades.
http::inaugural.senate.gov/2009

Inaugural Addresses of the Presidents of the United States
From George Washington's first address in 1789 to the present. Includes a note on the presidents who took the oath of office without a formal inauguration.
http://www.bartleby.com/124

**Figure 1: Aggregated content. Vertical content presented in a display above general web results.**

as a self-contained element of the search result page (Figure 1).

Vertical selection can be performed through hand-crafted rules or through machine learned query classifiers [3, 15]. These approaches represent queries in a high-dimensional feature space. Using a set of example queries, one can train a statistical model which predicts query intent given query features. By carefully choosing features, models can generalize from the training set to new queries. Features may include either lexical query features (e.g. query terms) or non-lexical query features (e.g. detected entity types).

Information retrieval systems, once in operation, can gather a large amount of user feedback which can be used to correct vertical selection errors. For example, in web search, measuring user activities such as clicks, timeouts, and reformulations can provide valuable—albeit noisy—feedback [20, 13]. Machine learned approaches (as discussed in the existing literature) would require manually mining new training data and periodically retraining a model with mined data. Even if retrained, due to the statistical and feature-based nature of these classifiers, there is no guarantee that future occurrences of new training queries will be correctly classified. As a result, system designers often need to heuristically construct 'fail-safe' dictionaries.

In this paper, we present models which combine a probabilistic query classifier (similar to those described in existing literature) with online user feedback. We will demonstrate that our methods significantly improve accuracy of intent detection. The generality of our problem definition and approaches present opportunities for further refinement as well as application to other federated search domains. Furthermore, we propose a simulation framework for evaluating vertical selection using parameters grounded in commercial query logs.

## 2. PREVIOUS WORK

Despite the increasing integration of vertical content with web search results, relatively few research papers address this topic. Previous work has addressed the integration of content from shopping [15], news [10], and job [15] verticals. These authors approach vertical selection as a binary classification task and evaluate performance on independent binary problems. Our work builds on these results by expanding the set of verticals and situating evaluation in a stream of queries with potentially multiple relevant verticals.

Distributed information retrieval refers to the task of generating a single ranked list from multiple sub-collections [6]. One important sub-task of distributed information retrieval, *resource selection*, refers to deciding which sub-collections to search given a user's query. If we consider verticals to be the sub-collections of interest, distributed information retrieval techniques can be directly applied to vertical selection. Our work extends resource selection by introducing user feedback and a simulated evaluation.

More generally, query classification refers to techniques for automatically matching queries to some predefined set of categories. Some query classification work has focused on classifying queries into topical categories, such as games, business, and health [23, 5, 15]. Our work extends query classification by firmly grounding categories in vertical content, introducing user feedback, and providing an evaluation situated in a search task.

Detecting and exploiting user feedback has been proposed for a variety of sub-tasks of web search. Joachims proposed using click-through data in order to learn good ranking functions [12]. Radlinski *et al.* use a multi-armed bandit algorithm to actively adapt retrieval results to gather feedback and improve ranking functions [19]. In the context of search advertising, approaches focus on explicitly modeling the click-through rate of query-advertisement pairs, in part because revenue is directly related to user clicks [21, 7]. Our work focuses on relevance rather than revenue and verticals rather than documents or advertisements.

Finally, our use of simulation for evaluation has precedent in document retrieval. The use of *deterministic simulation* underlies tasks such as relevance feedback, information filtering, and topic tracking; our work is more related to *parametric simulation.* Although Cooper first proposed a system for randomly simulating queries and documents [8], Griffiths was the first to estimate simulation parameters from empirical data [11]. Tague *et al.* simulated documents, queries, and relevance judgments using Cranfield II and Medlars corpora to estimate model parameters [25]. Parametric simulation has also been used to evaluate information filtering systems [17] and relevance feedback methods [26]. Azzopardi *et al.* used document language models to automatically create known-item relevance judgments [4]. Our work most closely resemble that of Acharya *et al.* who used a query and click simulator to evaluate web advertisement placement decisions [1]. Unlike these simulators, we ground the parameters of our model in a mix of both manually labeled data (in the spirit of deterministic simulation) and parametric simulation based on query log statistics.

## 3. PROBLEM OVERVIEW

Assume we have a stream of queries being submitted by users to the search engine. Our search results page always presents results from the web (i.e. "ten blue links"). In addition to a web index, the system also has access to $k$ verticals. The result of issuing the query to a vertical can be embedded in the search results page (e.g. Figure 1). In this work, we consider the situation where up to one vertical display can be integrated *above* web results. Reading the page from top to bottom, the user can be satisfied by the vertical content (if present) or the generic web results or neither. The user then communicates satisfaction to the system (we will elaborate on the nature of this feedback in Section 5.2). Our objective is to maximize the user satisfaction by presenting the appropriate vertical display; this includes the presentation of *no* display when the user is best satisfied by general web results.

## 4. ALGORITHM

In this section, we will describe a model which, given a query, predicts its vertical intent. The features of this model are non-lexical, allowing it to generalize beyond training queries. This model, once in operation, does not adapt to user feedback. Therefore, we refer to it as the offline model. In Section 4.2, we will describe several methods for combining user feedback and offline predictions. In Section 4.3, we present a method for incorporating information from related queries. Finally, in Section 4.4, we present a method for increasing robustness through randomization.

### 4.1 Vertical Selection with Offline Training

Our off-line model derives evidence from vertical content, vertical query logs, and the query string. Vertical content evidence includes classic distributed information retrieval and performance prediction metrics. Vertical query logs can be used to determine how similar a candidate query is to queries issued directly to the vertical. Finally, query string features include boolean evidence such as regular expressions and dictionaries believed to correlate with vertical intent. A more complete description of these features can be found in [3].

We would like to use these three classes of features in order to predict which vertical display—if any—to present to the user. Therefore, we have $k+1$ potential vertical intents, one for each vertical and one for "no relevant vertical". Because our features do not refer to the actual query tokens, we can expect new queries to be correctly predicted to the extent that there are correlations between these features and vertical intent.

Given a set of queries manually labeled with vertical intents, we can train a statistical model to predict the relevant verticals for new queries. In particular, we would like the model to compute the probability that a vertical is relevant given a query. We will model this as a set of $k+1$ Bernoulli random variables, each modeled in turn by a separate logistic regression. In our case, the explanatory variables are our features. If we train $k+1$ logistic regression models, then, given a query, we can predict the probability of success of each of the $k+1$ trials. For query $q$ and vertical $v$, let $\pi_q^v$ be the probability of that vertical being relevant as estimated by our logistic regression model. Given a query, the predicted vertical is $\mathrm{argmax}_v \pi_q^v$. This is equivalent to the technique of training $k+1$ one-versus-all binary classifiers often used for multiclass learning problem. Although we adopt logistic regression as our base classifier, any binary classifier can be used. Algorithms in subsequent sections only require that the classifier output a probability for each vertical.

## 4.2 Adaptation in the Presence of Feedback

Notice that the model in the previous section makes predictions only dependent on the query and does not change with user feedback. Assume we have some method for detecting the user satisfaction or dissatisfaction with the vertical display presented. In web search, satisfaction may be measured by a function of clicks or dwell time and dissatisfaction may be measured by a function of skips, reformulation, or abandonment [13, 20]. In this section, we will provide algorithms for adapting model predictions in the presence of *binary user feedback signals.* Although we leave the treatment of non-binary signals such as probabilities or real values for future work, we will address the robustness of our models to signal unreliability and noise in our evaluation.

### *4.2.1 Multiple Beta Prior*

If we assume that the relevance of each vertical can be represented as a Bernoulli random variable, then the conjugate prior is the Beta distribution. That is, we model each probability of relevance of a vertical to a query, $p_q^v$, as being sampled from a Beta distribution [10]. Mathematically,

$$p_q^v \sim \mathrm{Beta}(a_q^v, b_q^v)$$

The parameters $a_q^v$ and $b_q^v$ control the shape of our prior distribution over the probability that vertical $v$ is relevant. In practice, we use the offline model probability, $\pi_q^v$, in order to select these parameters,

$$a_q^v = \mu\pi_q^v \qquad\qquad b_q^v = \mu(1-\pi_q^v)$$

where $\mu$ is a hyperparameter of our model. A large value for $\mu$ will concentrate the distribution around $\pi_q^v$. A small value for $\mu$ will 'spread out' this distribution.

Assume we have positive and negative feedback information for a query-vertical pair. Let $\mathcal{R}_q^v$ be the number of positive impressions (e.g. clicks) and $\overline{\mathcal{R}}_q^v$ be the number of negative impressions (e.g. skips). Then, the posterior, given data from displays presented, is also a Beta distribution with a posterior mean of,

$$\tilde{p}_q^v = \frac{\mathcal{R}_q^v + \mu\pi_q^v}{\mathcal{V}_q^v + \mu} \tag{1}$$

where $\mathcal{V}_q^v = \mathcal{R}_q^v + \overline{\mathcal{R}}_q^v$ represents the number of times the vertical display $v$ was presented for query $q$. For small values of $\mu$, the model will be very sensitive to feedback from the user. For large values, the model will rely on $\pi_q^v$ more than feedback.

### *4.2.2 Logistic Normal Prior*

Alternatively, we can use a logistic normal prior to model user feedback [2]. In the context of information retrieval, this model was originally proposed for adaptation of document retrieval scores to user feedback [14]. We will briefly review the model and adapt notation for vertical selection. A more complete derivation can be found in references.

Assume that $t-1$ queries have been issued to the system and we wish to predict the vertical relevance for the next query. Define two $t \times k$ random matrices $\mathbf{W}$ and $\overline{\mathbf{W}}$. The elements of these matrices are sampled from a single multivariate normal,

$$\mathbf{W}, \overline{\mathbf{W}} \sim \mathcal{N}_{2tk}(\eta, \mathbf{\Sigma}) \tag{2}$$

where $\eta$ is a $2tk \times 1$ vector of means and $\mathbf{\Sigma}$ is a $2tk \times 2tk$ covariance matrix. At time $t$, define $p_q^v$ as,

$$p_q^v = \frac{\exp(W_{tv})}{\exp(W_{tv}) + \exp(\overline{W}_{tv})} \tag{3}$$

The prior mean at time $t$ can be written as a function of $\eta$ and $\mathbf{\Sigma}$. However, given covariances $\mathbf{\Sigma}$ we can select $\eta$ so that the prior mean matches a target value. In this work, we will define $\eta$ so that the prior means equal $\pi_q^v$.

The posterior mean after observing $t-1$ queries can be derived as,

$$\begin{aligned}
\tilde{p}_q^v &= \frac{\pi_q^v \exp(a_q^v)}{\pi_q^v \exp(a_q^v) + (1-\pi_q^v)\exp(b_q^v)} && (4)\\
a_q^v &= \sum_{i=1}^{t-1}\sum_{j=1}^{k} \mathcal{V}_i^j \left(\mathcal{R}_i^j \mathrm{Cov}(W_{tv}, W_{ij}) + \overline{\mathcal{R}}_i^j \mathrm{Cov}(W_{tv}, \overline{W}_{ij})\right)\\
b_q^v &= \sum_{i=1}^{t-1}\sum_{j=1}^{k} \mathcal{V}_i^j \left(\overline{\mathcal{R}}_i^j \mathrm{Cov}(\overline{W}_{tv}, \overline{W}_{ij}) + \mathcal{R}_i^j \mathrm{Cov}(\overline{W}_{tv}, W_{ij})\right)
\end{aligned}$$

where $\mathcal{V}_t^v$ is a boolean variable indicating whether vertical $v$ was presented at time $t$; $\mathcal{R}_t^v$ and $\overline{\mathcal{R}}_t^v$ are boolean variables indicating whether a vertical $v$ received positive or negative feedback at time $t$.

Notice that the estimate in Equation 4 only requires two columns of the covariance matrix, $\mathbf{\Sigma}$, those associated with $W_{tv}$ and $\overline{W}_{tv}$. We will define these covariances as,

$$\begin{aligned}
\mathrm{Cov}(W_{tv}, W_{ij}) &= \mathrm{Cov}(\overline{W}_{tv}, \overline{W}_{ij})\\
&= \begin{cases} 1 & \text{if } q_t = q_i \text{ and } v = j\\ 0 & \text{otherwise}\end{cases} && (5)\\
\mathrm{Cov}(W_{tv}, \overline{W}_{ij}) &= \mathrm{Cov}(\overline{W}_{tv}, W_{ij})\\
&= \begin{cases} \frac{\sigma}{\mathcal{V}_{q_i}^j} & \text{if } q_t = q_i \text{ and } v \neq j\\ 0 & \text{otherwise}\end{cases} && (6)
\end{aligned}$$

where $q_i$ is the query issued at time $i$. Equation 5 incorporates feedback from previous displays of $v$ for $q_t$. Equation 6 incorporates feedback from previous displays of competing verticals for $q_t$. The parameter $\sigma$ controls the *positive* contribution of *negative* feedback on competing verticals to a vertical's probability of relevance. When $\sigma$ is large, negative feedback on competing verticals positively affects a vertical's probability of relevance and *vice versa.* This naïvely assumes that all verticals are negatively correlated, a fact which is almost certainly untrue. However, we have empirically noticed that queries tend to have few ($< 3$) relevant verticals.

We use a relatively simple covariance function here. More elaborate covariance functions may incorporate temporal proximity for verticals with non-stationary relevance (e.g. news). In the next section, we will use this covariance matrix to encode information from related queries.

We can use Equations 5 and 6 in order to rewrite the exponents in Equation 4,

$$a^v_q = \mathcal{R}^v_q + \sum_{v' \neq v} \frac{\sigma}{\mathcal{V}^{v'}_{q'}} \overline{\mathcal{R}}^{v'}_q \qquad b^v_q = \overline{\mathcal{R}}^v_q + \sum_{v' \neq v} \frac{\sigma}{\mathcal{V}^{v'}_{q'}} \mathcal{R}^{v'}_q$$

## 4.3 Using Information from Similar Queries

We have demonstrated how to use feedback for a query to estimate $p^v_q$. However, a vertical's probability of being relevant is also likely to be related to the feedback on topically related queries. For example, consider a situation where we have seen many query events for "obama inauguration", providing a reliable estimate of $p^v_q$ for all $v$. Then, given the new query "barack obama inauguration", we have information about $p^v_{q'}$ if we know that these two queries are related.

There are several methods for detecting the similarity between queries. In this work, we adopt a corpus-based similarity measure using language models of retrieved results. Specifically, we create a language model by interpolating the top retrieved document language models weighted by their retrieval scores [9]. This provides a query language model for each query. Given two query language models, we compare two queries by comparing their associated language models using the Bhattacharyya correlation. The Bhattacharyya correlation ranges between 0 and 1 and is defined as,

$$\mathcal{B}(q_i, q_j) = \sum_{w \in \mathcal{V}} \sqrt{P(w|\theta_{q_i})P(w|\theta_{q_j})} \tag{7}$$

This approach was previously used for news vertical selection [10] and similar methods have been used for query suggestion and advertisement placement tasks [22, 16]. We note that while the Bhattacharyya similarity measure uses only $P(w|\theta_Q)$, other similarity measures based on time or term overlap can be used or combined with $\mathcal{B}$.

How we precisely incorporate information from similar queries depends on the prior we are using. For the multiple Beta model, our assumption is that if $\mathcal{B}(q, q')$ is large, then $p^v_q \approx p^v_{q'}$. We incorporate this information into the candidate query's prior. Specifically, define the nearest-neighbor estimate of $p^v_q$ as,

$$\hat{p}^v_q = \frac{1}{\mathcal{Z}_q} \sum_{q'} \mathcal{B}(q, q') \tilde{p}^v_{q'} \tag{8}$$

where $\mathcal{Z}_q = \sum_{q'} \mathcal{B}(q, q')$. We then compute a compound prior for $p^v_q$ as,

$$\hat{\pi}^v_q = (1 - \lambda_q)\pi^v_q + \lambda_q \hat{p}^v_q \tag{9}$$

where $\lambda \in [0, 1]$ controls the importance of the nearest-neighbor estimate relative to the offline model estimate. Notice that the normalization in Equation 8 discards the magnitude of the similarity. That is, the relative weights for $k$ queries with constant similarity $\forall q' : \mathcal{B}(q, q') = .9$ is the same as weights for $\forall q' \mathcal{B}(q, q') = .1$. In order to preserve the absolute confidence, we define $\lambda_q = \lambda \times \max_{q'} \mathcal{B}(q, q')$.

We incorporate similar queries in the logistic normal prior model by adding elements to $\mathbf{\Sigma}$. Assume for $q_i = q_{i'}$, covariances are described in Equations 5 and 6. For cases where $q_i \neq q_{i'}$,

$$\begin{aligned} \text{Cov}(W_{tv}, W_{ij}) &= \text{Cov}(\overline{W}_{tv}, \overline{W}_{ij}) \\ &= \begin{cases} \frac{\lambda \mathcal{B}(q_t, q_i)}{\mathcal{V}^v_{q_i}} & \text{if } v = j \\ 0 & \text{if } v \neq j \end{cases} \end{aligned} \tag{10}$$

$$\begin{aligned} \text{Cov}(W_{tv}, \overline{W}_{ij}) &= \text{Cov}(\overline{W}_{tv}, W_{ij}) \\ &= \begin{cases} 0 & \text{if } v = j \\ \frac{\lambda \sigma \mathcal{B}(q_t, q_i)}{\mathcal{V}^j_{q_i}} & \text{if } v \neq j \end{cases} \end{aligned} \tag{11}$$

where $\lambda$ controls the contribution from similar queries and $\sigma$ is defined in the previous section. We can use Equations 10 and 11 in order to rewrite the exponents in Equation 4,

$$\hat{a}^v_q = a^v_q + \lambda \sum_{q' \neq q} \mathcal{B}(q, q') \left( \frac{\mathcal{R}^v_{q'}}{\mathcal{V}^v_{q'}} + \sum_{v' \neq v} \frac{\sigma}{\mathcal{V}^{v'}_{q'}} \overline{\mathcal{R}}^{v'}_{q'} \right)$$

$$\hat{b}^v_q = b^v_q + \lambda \sum_{q' \neq q} \mathcal{B}(q, q') \left( \frac{\overline{\mathcal{R}}^v_{q'}}{\mathcal{V}^v_{q'}} + \sum_{v' \neq v} \frac{\sigma}{\mathcal{V}^{v'}_{q'}} \mathcal{R}^{v'}_{q'} \right)$$

## 4.4 Randomizing Decisions

Sometimes, we would like to present a vertical display for a query even though it is not predicted to be the display with the highest probability. The motivation for this may be to gather a small amount of feedback without devastating system performance. To this end, we can present random displays for queries with some probability, $\epsilon$. This approach is referred to as the $\epsilon$-greedy approach [24].

The $\epsilon$-greedy approach aggressively presents verticals regardless of what we have learned about the vertical probabilities. We can incorporate this knowledge by exploiting the posterior means across verticals. Specifically, we will sample our decision from a multinomial over verticals derived from our estimated vertical relevance probabilities, $\tilde{p}^v_q$. In our experiments, we use a Boltzmann distribution to construct the multinomial,

$$P(v) = \frac{1}{\mathcal{Z}} \exp\left(\frac{\tilde{p}^v_q}{\tau}\right) \tag{12}$$

where $\mathcal{Z} = \sum_v \exp\left(\frac{\tilde{p}^v_q}{\tau}\right)$ and $\tau > 0$ controls the uniformity of the random vertical selection. As $\tau \to \infty$, the vertical selection becomes more random. At $\tau \to 0$, the vertical selection becomes greedy. In reinforcement learning, state space exploration is often performed using a Boltzmann distribution [24].

## 5. METHODS AND MATERIALS

## 5.1 Data

Our data set consists of 25195 queries sampled from a web search query log. Human editors were provided a list of 18 verticals (Table 1) and asked to assign from zero to six relevant verticals for each query. Queries were automatically spell corrected and normalized by down-casing and removing punctuation. The vertical distribution is shown in Table 1. About 26% of queries were assigned "no relevant vertical", indicating that these were best served by general web content. Of those assigned at least one relevant vertical, the average number of verticals assigned per query was 1.49. Of these, 60% were assigned one vertical, 31% were assigned

| | | | |
|---|---|---|---|
| autos | 3.0% | music | 4.6% |
| directory | 4.4% | news | 5.1% |
| finance | 2.6% | reference | 15.4% |
| games | 2.6% | shopping | 20.3% |
| health | 4.3% | sports | 3.3% |
| jobs | 1.5% | travel | 8.7% |
| image | 6.0% | tv | 2.7% |
| local | 19.1% | video | 3.1% |
| maps | 1.1% | no relevant vertical | 26.3% |
| movies | 2.3% | | |

**Table 1: Percentage of queries assigned each vertical. Percentages do not sum to one because queries can be assigned more than one relevant vertical.**

two verticals, and 9% were assigned more than two. We trained our offline model using 10 fold cross-validation on all of the labeled queries. We used a snapshot of Wikipedia as the corpus to compute query similarity.

## 5.2 Query Simulation

Because we are interested in the adaptation of an offline model to online feedback, we used our offline data to construct a query simulator. First, we retrospectively analyzed the query frequency of each of the queries in our data set. We then estimated the parameters of a multinomial over our 25195 queries using the relative frequency data; call this multinomial $\theta_Q$. We found that this distribution was skewed and heavy-tailed, a property common in many real query logs.

We would like to simulate a user submitting a query and having a particular intent chosen from those listed in Table 1. In the case of "no relevant vertical", we assume that the user is satisfied by general web results and, therefore we refer to this as the "Web" vertical. In order to simulate the generation of vertical intent, for each query, $q$, we estimate a multinomial over verticals using the manual judgements with uniform probability over relevant verticals and zero probability for non-relevant verticals; call this multinomial $\theta_V^q$.

The simulation first samples a query, $q_t$, from $\theta_Q$ and then one of $q_t$'s relevant verticals from $\theta_V^{q_t}$. We repeat this process for 10 million query samples, evaluating performance at the end of the simulation. We present average and standard deviations of metrics over 10 runs of the simulator. We found 10 runs to be sufficient to achieve statistically different performances in algorithms.

So far, our simulated user provides perfect feedback. If the relevant vertical is displayed, she provides positive feedback; if a non-relevant vertical is displayed, she provides negative feedback. In practice, measuring user feedback is imperfect and signals are noisy. If a user is satisfied, sometimes we incorrectly detect negative feedback; if a user is unsatisfied, sometimes we incorrectly detect positive feedback. At other times, the system may have predicted the correct intent but the interface may have resulted in an incorrect user response.

Noisy feedback influences two aspects of system development: adaptation and evaluation. In the case of adaptation, although we expect performance to decrease in the presence of noisy feedback, we prefer adaptation algorithms which are robust to noisy feedback. In the case of evaluation, noisy feedback can misrepresent performance. This is one drawback we see with relevance metrics based exclusively on noisily detected user feedback.

Using a simulation allows us to decouple feedback signals and the 'true' user feedback variable. Because we know the true vertical intents, we can evaluate using a noiseless signal while introducing noise only in the feedback. We introduce noise by defining a probability, $\delta$, of correctly detecting user feedback. When a correct display is presented, the system detects positive feedback with this probability; similarly, it detects negative feedback with this probability. The simulation is more formally defined in Appendix A.

## 5.3 Evaluation

Assume we have a stream of queries indexed by $t$ where $y_t \in \mathcal{V} \cup \{\text{Web}\}$ is the user's intent. The system's prediction is $f_t \in \mathcal{V} \cup \{\text{Web}\}$. We represent the user's utility from the presentation as,

$$u(y_t, f_t) = \begin{cases} 1 & y_t = f_t \\ \alpha & (y_t = \text{Web}) \wedge (f_t \neq \text{Web}) \\ 0 & \text{otherwise} \end{cases}$$

where $0 \leq \alpha \leq 1$ represents the user's discounted utility by being presented a display above the desired web results. When $\alpha$ is large, a user's satisfaction is not affected by injecting the display above the desired web results. When $\alpha$ is small, a user's satisfaction is degraded. When $\alpha = 0$, a user is so distracted by the display that there is no satisfaction by the web results. In our experiments, we use $\alpha = \frac{1}{2}$ which is similar to the discount often used in measures such as NDCG.

For an individual query, we compute the average utility as,

$$\text{U}_q = \frac{1}{|T_q|} \sum_{t \in T_q} u(y_t, f_t)$$

where $T_q$ is the set of times query $q$ was issued. Over all queries, we define the macro-averaged utility as

$$\text{U}_{\text{macro}} = \frac{1}{|Q|} \sum_{q \in Q} \text{U}_q \tag{13}$$

We adopt a macro-averaged metric because a micro-averaged metric would be dominated by relatively few queries.

About 30% of our queries are labeled with more than one relevant vertical. For these queries, our simulator selects one relevant vertical from the relevant set at each step. Because we do not condition the sampling on any variable, the best a system can do is to either learn one of the relevant verticals and select that vertical every time or learn all of the relevant verticals and randomly select from them. Both approaches converge to an average utility less than one. For our collection, the macro-averaged utility of such a selector over all queries is expected to be 0.843. Therefore, we normalize Equation 13 by this factor. The macro-averaged utility over queries with multiple relevant verticals is expected to be 0.461. When evaluating only on queries with multiple intents, we normalize by this factor.

## 6. RESULTS

In our discussion, we will refer to the multiple Beta model as MB and the logistic normal model as LN. Superscripts indicate the prior used by the model. The prior $\pi$ refers to the model described in Section 4.1. The prior $\mathcal{U}$ refers to a uniform prior which assigns $\pi_q^v = \frac{1}{2}$ for all query-vertical

| $\mu$ | $\delta = 0.95$ | $\delta = 0.90$ | $\delta = 0.75$ |
|---|---|---|---|
| 0.10 | $0.872 \pm 0.002$ | $0.819 \pm 0.001$ | $0.679 \pm 0.003$ |
| 0.25 | $\mathbf{0.878 \pm 0.002}$ | $0.832 \pm 0.002$ | $0.700 \pm 0.002$ |
| 0.50 | $0.877 \pm 0.002$ | $\mathbf{0.836 \pm 0.001}$ | $0.718 \pm 0.002$ |
| 0.75 | $0.873 \pm 0.001$ | $0.834 \pm 0.002$ | $0.726 \pm 0.002$ |
| 0.90 | $0.871 \pm 0.001$ | $0.833 \pm 0.001$ | $0.730 \pm 0.002$ |
| 1.00 | $0.870 \pm 0.001$ | $0.832 \pm 0.001$ | $0.731 \pm 0.001$ |
| 2.00 | $0.858 \pm 0.001$ | $0.822 \pm 0.001$ | $\mathbf{0.733 \pm 0.001}$ |
| 3.00 | $0.848 \pm 0.001$ | $0.814 \pm 0.002$ | $0.731 \pm 0.002$ |
| 4.00 | $0.841 \pm 0.001$ | $0.808 \pm 0.002$ | $0.728 \pm 0.001$ |
| 5.00 | $0.836 \pm 0.001$ | $0.802 \pm 0.001$ | $0.726 \pm 0.001$ |

**Table 2: Normalized $\mathbf{U_{macro}}$ for $\mathbf{MB^{\pi}}$ runs.**

| $\sigma$ | $\delta = 0.95$ | $\delta = 0.90$ | $\delta = 0.75$ |
|---|---|---|---|
| 0.10 | $0.886 \pm 0.001$ | $0.880 \pm 0.001$ | $\mathbf{0.851 \pm 0.002}$ |
| 0.20 | $0.887 \pm 0.001$ | $0.880 \pm 0.001$ | $0.851 \pm 0.001$ |
| 0.30 | $0.887 \pm 0.001$ | $0.880 \pm 0.001$ | $0.849 \pm 0.001$ |
| 0.40 | $0.888 \pm 0.001$ | $0.881 \pm 0.001$ | $0.849 \pm 0.001$ |
| 0.50 | $0.888 \pm 0.001$ | $0.881 \pm 0.001$ | $0.849 \pm 0.001$ |
| 0.60 | $0.890 \pm 0.001$ | $0.883 \pm 0.001$ | $0.848 \pm 0.001$ |
| 0.70 | $0.890 \pm 0.001$ | $0.883 \pm 0.001$ | $0.848 \pm 0.001$ |
| 0.80 | $0.890 \pm 0.001$ | $0.883 \pm 0.001$ | $0.848 \pm 0.001$ |
| 0.90 | $0.891 \pm 0.001$ | $0.883 \pm 0.001$ | $0.848 \pm 0.001$ |
| 1.00 | $\mathbf{0.891 \pm 0.002}$ | $\mathbf{0.883 \pm 0.001}$ | $0.848 \pm 0.001$ |

**Table 3: Normalized $\mathbf{U_{macro}}$ for $\mathbf{LN^{\pi}}$ runs.**

pairs. The sub-script $s$ indicates the use of similar queries. Finally, the prefix indicates the randomization method used, $\epsilon$ for $\epsilon$-greedy and 'B' for Boltzmann.

We begin by comparing the baseline predictor, $\pi$, with those that use feedback information. The static baseline achieves $U_{macro}$ of $0.618 \pm 0.001$. In Table 2, we present the performance of $MB^{\pi}$ for various settings of $\mu$. We observe that, for all values of $\mu$, we improve over the baseline. In Table 3, we show performance of $LN^{\pi}$ for various values of $\sigma$. Again, for all values of $\sigma$, we improve over the baseline. While being able to improve using feedback is not surprising, we note that $LN^{\pi}$ always outperforms $MB^{\pi}$, especially as feedback becomes noisier. We can explain this behavior by inspecting the computation of the posterior mean for each prior. As evidence is introduced, the posterior for $MB^{\pi}$ (Equation 1) converges toward the true mean. The posterior for $LN^{\pi}$ (Equation 4) converges toward 0 or 1 with relatively little feedback. The negative correlations between competing verticals in $LN^{\pi}$ further concentrates probability in a few verticals. For example, assume there is one relevant vertical for a query. Further, assume that our prior is 0.90 for the relevant vertical and 0.85 for a non-relevant vertical. If $\delta = 0.75$, then our posterior using $MB^{\pi}$ converges to 0.75 for the relevant vertical. The system then has to also expend interactions to drive the non-relevant vertical's posterior below 0.75. $LN^{\pi}$, however, pushes queries towards extremes and therefore more gracefully handles noise when evaluating over all queries.

We performed experiments using a uniform prior (table suppressed due to space constraints). In these experiments, the best performing setting of $\mu$ for $MB^{\mathcal{U}}$ resulted in an average utility of $0.745 \pm 0.001$ ($\delta = 0.95$), $0.732 \pm 0.001$ ($\delta = 0.90$), and $0.669 \pm 0.001$ ($\delta = 0.75$). Observing that this outperforms the baseline, $\pi$, at all values of $\delta$, we may be tempted to conclude that the training of the logistic regression models is unnecessary. However, the combination of feedback and the offline model significantly outperforms the uniform prior. We observed similar results for $LN^{\mathcal{U}}$.

| $\lambda$ | $\delta = 0.95$ | $\delta = 0.90$ | $\delta = 0.75$ |
|---|---|---|---|
| 0.10 | $0.879 \pm 0.001$ | $0.837 \pm 0.002$ | $\mathbf{0.718 \pm 0.002}$ |
| 0.25 | $0.883 \pm 0.001$ | $0.841 \pm 0.001$ | $0.715 \pm 0.002$ |
| 0.50 | $\mathbf{0.884 \pm 0.001}$ | $\mathbf{0.842 \pm 0.002}$ | $0.716 \pm 0.001$ |
| 0.75 | $0.884 \pm 0.002$ | $0.841 \pm 0.001$ | $0.708 \pm 0.001$ |
| 0.90 | $0.883 \pm 0.002$ | $0.839 \pm 0.001$ | $0.705 \pm 0.002$ |

**Table 4: Normalized $\mathbf{U_{macro}}$ for $\mathbf{MB_s^{\pi}}$ runs.**

| $\tau$ | $\delta = 0.95$ | $\delta = 0.90$ | $\delta = 0.75$ |
|---|---|---|---|
| 0.005 | $0.880 \pm 0.001$ | $0.839 \pm 0.001$ | $0.733 \pm 0.001$ |
| 0.010 | $0.888 \pm 0.001$ | $0.853 \pm 0.001$ | $0.738 \pm 0.002$ |
| 0.025 | $\mathbf{0.896 \pm 0.001}$ | $0.877 \pm 0.001$ | $0.773 \pm 0.002$ |
| 0.050 | $0.892 \pm 0.001$ | $\mathbf{0.881 \pm 0.001}$ | $\mathbf{0.813 \pm 0.001}$ |
| 0.075 | $0.881 \pm 0.002$ | $0.867 \pm 0.001$ | $0.804 \pm 0.001$ |

**Table 5: Normalized $\mathbf{U_{macro}}$ for B-$\mathbf{MB^{\pi}}$ runs. This table should be compared with Table 2.**

The performance improvement from using similar queries is significant but slight (Table 4). When more feedback noise is introduced, gains disappear altogether. This suggests that similar queries are only useful to the extent that their estimates are accurate. With very noisy feedback, we gain very little information from related queries. We observed similar results for $LN_s^{\pi}$.

The improvement from decision randomization depends on the model prior. Boltzmann randomization significantly improves $MB^{\pi}$ (Table 5) while not affecting $LN^{\pi}$ (Table 6). We alluded to the reason B-$MB^{\pi}$ improves performance over $MB^{\pi}$ earlier. Namely, randomization allows the system to simultaneously adjust estimates toward the true mean. $LN^{\pi}$, however, quickly converges toward the extreme and therefore does not require simultaneous adjustment.

We present the best performing runs for all of our algorithms in Table 7. We find that, across values of $\delta$, $LN_s^{\pi}$ tends to perform best. However, similarity information does not appear to be useful when $\delta = 0.75$. If we use $MB^{\pi}$, then randomized decision making performs best.

Something interesting happens if we inspect performance only for queries with multiple relevant verticals (Table 8). In this situation, the simulator will randomly select between relevant verticals. Notice that for MB and LN methods individually, the effects are the same as those observed for all queries. However, relative performance between the two priors switches for multiple intent queries. This behavior results from the aggressive use of feedback in $LN^{\pi}$ which is exponential in the amount of feedback. In the cases where feedback is split between two relevant verticals, posterior distributions of relevant verticals will swing from 0 to 1 given inconsistent vertical feedback.

## 7. CONCLUSIONS AND FUTURE WORK

We have presented several algorithms for combining user

| $\tau$ | $\delta = 0.95$ | $\delta = 0.90$ | $\delta = 0.75$ |
|---|---|---|---|
| 0.005 | $0.883 \pm 0.001$ | $0.877 \pm 0.001$ | $0.843 \pm 0.002$ |
| 0.010 | $0.881 \pm 0.001$ | $0.875 \pm 0.001$ | $0.843 \pm 0.001$ |
| 0.025 | $0.881 \pm 0.001$ | $0.873 \pm 0.001$ | $0.841 \pm 0.001$ |
| 0.050 | $0.877 \pm 0.001$ | $0.870 \pm 0.001$ | $0.836 \pm 0.001$ |
| 0.075 | $0.867 \pm 0.001$ | $0.861 \pm 0.001$ | $0.828 \pm 0.001$ |

**Table 6: Normalized $\mathbf{U_{macro}}$ for B-$\mathbf{LN^{\pi}}$ runs. This table should be compared with Table 3.**

|  | $\delta = 0.95$ | $\delta = 0.90$ | $\delta = 0.75$ |
|---|---|---|---|
| $\pi$ | $0.618 \pm 0.001$ | $0.618 \pm 0.001$ | $0.618 \pm 0.001$ |
| $\text{MB}^{\mathcal{U}}$ | $0.745 \pm 0.001$ | $0.732 \pm 0.001$ | $0.669 \pm 0.001$ |
| $\text{MB}^{\pi}$ | $0.878 \pm 0.002$ | $0.836 \pm 0.001$ | $0.733 \pm 0.001$ |
| $\text{MB}^{\pi}_{s}$ | $0.885 \pm 0.002$ | $0.843 \pm 0.002$ | $0.730 \pm 0.003$ |
| $\epsilon\text{-MB}^{\pi}$ | $0.870 \pm 0.001$ | $0.835 \pm 0.002$ | $0.752 \pm 0.001$ |
| $\text{B-MB}^{\pi}$ | $\mathbf{0.896 \pm 0.001}$ | $0.881 \pm 0.001$ | $0.816 \pm 0.001$ |
| $\text{LN}^{\mathcal{U}}$ | $0.722 \pm 0.001$ | $0.709 \pm 0.001$ | $0.650 \pm 0.001$ |
| $\text{LN}^{\pi}$ | $0.891 \pm 0.002$ | $0.883 \pm 0.001$ | $0.851 \pm 0.001$ |
| $\text{LN}^{\pi}_{s}$ | $0.894 \pm 0.002$ | $\mathbf{0.887 \pm 0.002}$ | $\mathbf{0.853 \pm 0.002}$ |
| $\epsilon\text{-LN}^{\pi}$ | $0.891 \pm 0.001$ | $0.883 \pm 0.001$ | $0.851 \pm 0.001$ |
| $\text{B-LN}^{\pi}$ | $0.887 \pm 0.001$ | $0.880 \pm 0.001$ | $0.847 \pm 0.001$ |

**Table 7: Normalized $\mathbf{U_{macro}}$ for all queries. Performance using optimal parameter settings.**

|  | $\delta = 0.95$ | $\delta = 0.90$ | $\delta = 0.75$ |
|---|---|---|---|
| $\pi$ | $0.681 \pm 0.002$ | $0.681 \pm 0.002$ | $0.681 \pm 0.002$ |
| $\text{MB}^{\mathcal{U}}$ | $0.657 \pm 0.002$ | $0.636 \pm 0.002$ | $0.549 \pm 0.001$ |
| $\text{MB}^{\pi}$ | $0.883 \pm 0.002$ | $0.846 \pm 0.002$ | $0.744 \pm 0.002$ |
| $\text{MB}^{\pi}_{s}$ | $0.896 \pm 0.004$ | $0.855 \pm 0.002$ | $0.744 \pm 0.003$ |
| $\epsilon\text{-MB}^{\pi}$ | $0.885 \pm 0.004$ | $0.846 \pm 0.002$ | $0.748 \pm 0.002$ |
| $\text{B-MB}^{\pi}$ | $\mathbf{0.907 \pm 0.003}$ | $\mathbf{0.889 \pm 0.003}$ | $\mathbf{0.826 \pm 0.003}$ |
| $\text{LN}^{\mathcal{U}}$ | $0.510 \pm 0.001$ | $0.492 \pm 0.002$ | $0.421 \pm 0.002$ |
| $\text{LN}^{\pi}$ | $0.781 \pm 0.003$ | $0.772 \pm 0.002$ | $0.727 \pm 0.003$ |
| $\text{LN}^{\pi}_{s}$ | $0.796 \pm 0.003$ | $0.785 \pm 0.003$ | $0.735 \pm 0.003$ |
| $\epsilon\text{-LN}^{\pi}$ | $0.781 \pm 0.004$ | $0.772 \pm 0.003$ | $0.727 \pm 0.003$ |
| $\text{B-LN}^{\pi}$ | $0.755 \pm 0.002$ | $0.748 \pm 0.003$ | $0.701 \pm 0.002$ |

**Table 8: Normalized $\mathbf{U_{macro}}$ for multiple intent queries.**

feedback with offline classifier information. Our results demonstrate that, although feedback-only models can outperform offline-only models, combining the two results in significant improvements. We found that using a logistic normal prior outperforms using multiple beta priors across all queries. However, we also found that the multiple beta priors with randomized decision making provides stable performance for both single and multiple intent queries. Finally, we presented a query simulation and vertical feedback model which decouples evaluation signals and feedback signals, allowing us to accurately measure model robustness to feedback detection noise.

There are several directions in which to extend this work. First, we assumed that a search results page could only include a single vertical display. In practice, multiple displays can be stacked above the web results or blended elsewhere in the ranked list. Second, we have attempted to define and evaluate our algorithms with generalization in mind. We believe there is merit in applying these algorithms to other distributed information retrieval and federated search tasks. Third, we are interested in exploring domains where our simulator assumptions—in particular with respect to query and vertical distributions—do not hold. For example, we have introduced a fixed noise parameter, assuming that we detect positive and negative feedback equally well and that we detect feedback for different vertical displays equally well. In reality these noise rates are likely to be dependent on the type of feedback and vertical. Fourth, we also would like to modify our algorithms to deal with non-binary feedback, such as that statistically detected using implicit feedback features. Finally, although we believe that our simulation provides an attractive, noiseless evaluation, we acknowledge that the efficacy of these algorithms can only be confirmed in real query traffic. It is important, though, to be cautious about non-uniform noise in the evaluation signal, be it clicks or some other measurement.

## 8. ACKNOWLEDGMENTS

We would like to thank Jean François Beaumont, Daniel Boies, Hugues Bouchard, Jean François Crespo, Deborah Donato, Rosie Jones, Remi Kwan, Vanessa Murdock, Jian-Yun Nie, Jean François Paiement, and Alexandre Rochette for helpful discussions and feedback. This work was supported in part by the NSF grant IIS-0841275 and a generous gift from Yahoo!. Any opinions, findings, conclusions, and recommendations expressed in this paper are the authors' and do not necessarily reflect those of the sponsors.

## 9. REFERENCES

[1] S. Acharya, P. Krishnamurthy, K. Deshpande, T. Yan, and C.-C. Chang. A simulation framework for evaluating designs for sponsored search markets. In *WWW 2007 Workshop on Sponsored Search Auctions*, 2007.

[2] J. Aitchison and S. M. Shen. Logistic-normal distributions: Some properties and uses. *Biometrika*, 67(2):261–272, August 1980.

[3] J. Arguello, F. Diaz, J. Callan, and J.-F. Crespo. Sources of evidence for vertical selection. In *SIGIR 2009*, 2009.

[4] L. Azzopardi, M. de Rijke, and K. Balog. Building simulated queries for known-item topics: an analysis using six european languages. In *SIGIR 2007*, pages 455–462, 2007.

[5] S. M. Beitzel, E. C. Jensen, O. Frieder, D. D. Lewis, A. Chowdhury, and A. Kolcz. Improving automatic query classification via semi-supervised learning. In *ICDM 2005*, pages 42–49, 2005.

[6] J. Callan. Distributed information retrieval. In W. B. Croft, editor, *Advances in Information Retrieval*. 2000.

[7] M. Ciaramita, V. Murdock, and V. Plachouras. Online learning from click data for sponsored search. In *WWW 2008*, pages 227–236, 2008.

[8] M. D. Cooper. A simulation model of an information retrieval system. *Information Storage and Retrieval*, 9(1):13–32, 1973.

[9] S. Cronen-Townsend, Y. Zhou, and W. B. Croft. Predicting query performance. In *SIGIR 2002*, pages 299–306, 2002.

[10] F. Diaz. Integration of news content into web results. In *WSDM 2009*, 2009.

[11] J.-M. Griffiths. *The computer simulation of information retrieval systems*. PhD thesis, University College London, 1977.

[12] T. Joachims. Optimizing search engines using clickthrough data. In *KDD 2002*, pages 133–142, 2002.

[13] R. Jones and K. L. Klinkner. Beyond the session timeout: automatic hierarchical segmentation of search topics in query logs. In *CIKM 2008*, pages 699–708, 2008.

[14] P. J. Lenk and B. D. Floyd. Dynamically updating relevance judgements in probabilistic information systems via users' feedback. *Management Science*, 34(12):1450–1459, December 1988.

[15] X. Li, Y.-Y. Wang, and A. Acero. Learning query intent from regularized click graphs. In *SIGIR 2008*, pages 339–346, 2008.

[16] D. Metzler, S. T. Dumais, and C. Meek. Similarity measures for short segments of text. In *ECIR 2007*, pages 16–27, 2007.

[17] J. Mostafa, S. Mukhopadhyay, and M. Palakal. Simulation studies of different dimensions of users' interests and their

impact on user modeling and information filtering. *Information Retrieval*, 6:199–223, April 2003.
[18] V. Murdock and M. Lalmas, editors. *Proceedings of the SIGIR Workshop on Aggregated Search*, 2008.
[19] F. Radlinski, R. Kleinberg, and T. Joachims. Learning diverse rankings with multi-armed bandits. In *ICML 2008*, pages 784–791, 2008.
[20] F. Radlinski, M. Kurup, and T. Joachims. How does clickthrough data reflect retrieval quality? In *CIKM 2008*, pages 43–52, 2008.
[21] M. Richardson, E. Dominowska, and R. Ragno. Predicting clicks: estimating the click-through rate for new ads. In *WWW 2007*, pages 521–530, 2007.
[22] M. Sahami and T. D. Heilman. A web-based kernel function for measuring the similarity of short text snippets. In *WWW 2006*, pages 377–386, 2006.
[23] D. Shen, J.-T. Sun, Q. Yang, and Z. Chen. Building bridges for web query classification. In *SIGIR 2006*, pages 131–138, 2006.
[24] R. Sutton and A. Barto. *Reinforcement Learning.* 1998.
[25] J. Tague, M. Nelson, and H. Wu. Problems in the simulation of bibliographic retrieval systems. In *SIGIR 1980*, pages 236–255, 1980.
[26] R. W. White, I. Ruthven, J. M. Jose, and C. J. van Rijsbergen. Evaluating implicit feedback models using searcher simulations. *TOIS*, 23(3):325–361, July 2005.

## APPENDIX

## A. QUERY SIMULATOR

Our simulator is comprised of two parts: the intent simulator and the feedback detector. At time $t$, the intent simulator (Figure 2) first selects a labeled query, $q_t$, from the multinomial over queries, $\theta_Q$, (Line 2) and then a relevant vertical from the query-specific multinomial over verticals, $\theta_V^{q_t}$ (Line 3). The retrieval system is asked to make a vertical prediction for $q_t$ (Line 4). We evaluate the utility gain using the true relevant vertical and the prediction (Lines 5-6). Unlike click-based evaluation, there is no noise in evaluation.

The feedback detector (Figure 3) allows the experimenter to introduce noise into the system's feedback detection using the parameter $\delta$. Our simulated user reads the results page (Figure 1) from top to bottom. If the user is satisfied by the displayed vertical, we provide positive feedback with probability $\delta$ (Lines 4-6); if the user is not satisfied by the displayed vertical, we provide positive feedback with probability $1 - \delta$ (Lines 7-9). If the system presented a vertical display and the simulated user did not provide positive feedback on the display, then this implies negative feedback on the display (Line 10). This appeals in spirit to Joachim's work in preference mining from clicks [12]. In the event that the display did not receive feedback, the simulated user potentially provides positive feedback on the general web results (Lines 14-21); the logic follows the logic in Lines 3-10.

```
SIMULATE-QUERIES($T, \delta$)
 1  $t \leftarrow 0$
 2  repeat  $q_t \sim \theta_Q$                               ▷ sample query
 3          $y_t \sim \theta_V^{q_t}$                        ▷ sample vertical
 4          $f_t \leftarrow f(q_t)$                          ▷ make prediction

 5          $g[q_t] \leftarrow g[q_t] + u(y_t, f_t)$         ▷ utility gain
 6          $n[q_t] \leftarrow n[q_t] + 1$                   ▷ query count

 7          SIMULATE-NOISY-FEEDBACK($f_t, y_t, \delta$)

 8          $t \leftarrow t + 1$
 9    until $t = T$

10  $\mathrm{U_{macro}} \leftarrow \frac{1}{|Q|} \sum_{q \in Q} (g/n)[q]$
```

**Figure 2: Query simulator pseudocode.**

```
SIMULATE-NOISY-FEEDBACK($f_t, y_t, \delta$)
 1  $r \sim \mathcal{U}(0,1)$
 2  $n \sim \mathcal{U}(0,1)$

 3  $\mathcal{V}_q^{f_t} \leftarrow \mathcal{V}_q^{f_t} + 1$                    ▷ increment view

 4  if $(f_t = y_t) \wedge (r < \delta)$
 5     then $\mathcal{R}_q^{f_t} \leftarrow \mathcal{R}_q^{f_t} + 1$          ▷ correct pos. feedback
 6          return

 7  if $(f_t \neq y_t) \wedge (n > \delta)$
 8     then $\mathcal{R}_q^{f_t} \leftarrow \mathcal{R}_q^{f_t} + 1$          ▷ incorrect pos. feedback
 9          return

10  $\overline{\mathcal{R}}_q^{f_t} \leftarrow \overline{\mathcal{R}}_q^{f_t} + 1$   ▷ neg. feedback

11  if $f_t \neq$ Web                                   ▷ a display was presented

12     then $r \sim \mathcal{U}(0,1)$
13          $n \sim \mathcal{U}(0,1)$

14          $\mathcal{V}_q^{\mathrm{Web}} \leftarrow \mathcal{V}_q^{\mathrm{Web}} + 1$        ▷ increment Web view

15          if $(y_t = \mathrm{Web}) \wedge (r < \delta)$
16            then $\mathcal{R}_q^{\mathrm{Web}} \leftarrow \mathcal{R}_q^{\mathrm{Web}} + 1$   ▷ correct pos.
17                 return                                  ▷ feedback

18          if $(y_t \neq \mathrm{Web}) \wedge (n > \delta)$
19            then $\mathcal{R}_q^{\mathrm{Web}} \leftarrow \mathcal{R}_q^{\mathrm{Web}} + 1$   ▷ incorrect pos.
20                 return                                  ▷ feedback

21          $\overline{\mathcal{R}}_q^{\mathrm{Web}} \leftarrow \overline{\mathcal{R}}_q^{\mathrm{Web}} + 1$   ▷ neg. feedback
```

**Figure 3: Noisy feedback simulator pseudocode.**