A Unified Search Federation System Based on Online User Feedback

Luo Jie
Yahoo! Labs
701 First Ave.
Sunnyvale, CA 94089
luojie@yahoo-inc.com

Sudarshan Lamkhede
Yahoo! Labs
701 First Ave.
Sunnyvale, CA 94089
lamkhede@yahoo-inc.com

Rochit Sapra
Yahoo! Search
701 First Ave.
Sunnyvale, CA 94089
rsapra@yahoo-inc.com

Evans Hsu
Yahoo! Taiwan
Nangang Science Park III
San-Chung Rd
Nangang Dist. 115, Taiwan
evanshsu@yahoo-inc.com

Helen Song
Yahoo! Search
701 First Ave.
Sunnyvale, CA 94089
leis@yahoo-inc.com

Yi Chang
Yahoo! Labs
701 First Ave.
Sunnyvale, CA 94089
yichang@yahoo-inc.com

ABSTRACT

Today's popular web search engines expand the search process beyond crawled web pages to specialized corpora ("verticals") like images, videos, news, local, sports, finance, and shopping etc., each with its own specialized search engine. Search federation deals with problems of the selection of search engines to query and merging of their results into a single result set. Despite a few recent advances, the problem is still very challenging. First, due to the heterogeneous nature of different verticals, how the system merges the vertical results with the web documents to serve the user's information need is still an open problem. Moreover, the scale of the search engine and the increasing number of vertical properties requires a solution which is efficient and scaleable. In this paper, we propose an unified framework for the search federation problem. We model the search federation as a contextual bandit problem. The system uses reward as a proxy for user satisfaction. Given a query, our system predicts the expected reward for each vertical, then organizes the search result page (SERP) in a way which maximizes the total reward. Instead of relying on human judges, our system leverages implicit user feedback to learn the model. The method is efficient to implement and can be applied to verticals of different nature. We have successfully deployed the system to three different markets, and it handles multiple verticals in each market. The system is now serving hundreds of millions of queries live each day, and has improved user metrics considerably.

KDD'13, August 11–14, 2013, Chicago, Illinois, USA.

Categories and Subject Descriptors

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

General Terms

Algorithms, Experimentation, System

Keywords

federated search, user feedback, machine learning

1. INTRODUCTION

Modern search engines have gone beyond presenting only web text documents. They have extended their services to include results from specialized corpora or *verticals* likes news, local search, movie, shopping etc. When a user issues a query, the search engine sends it to different vertical search engines, and the results returned by these engines are then aggregated together with the web (algo) results and composed into a search results page (SERP), as the example shown in Figure 1. This process is usually referred as *aggregated search* or *federated search*. A good search federation system can help enhancing the user's search experience. The results from the verticals are presented to the users in a visually appealing interface and contain useful information. For example, a movie result usually contains the movie poster and showtime information, and a local search result contains the address and telephone of the target business.

Given a user query, the system should first decide the possible vertical intent(s) of the query and send it to the corresponding vertical backend(s). Some queries may express the intent for vertical content explicitly, (e.g., "election news" and "news about sandy"), other queries' intent maybe implicit (e.g., "election" and "sandy"). Moreover, some specific queries may contain multiple intents. A user who searches for "coffee bean" may look for a local shop which sells coffee beans (local intent), plan to buy coffee beans online (shopping intent), or search for an article which recommends coffee beans (web articles). Therefore, how to compare the relevance of these items, and place the verticals in the correct

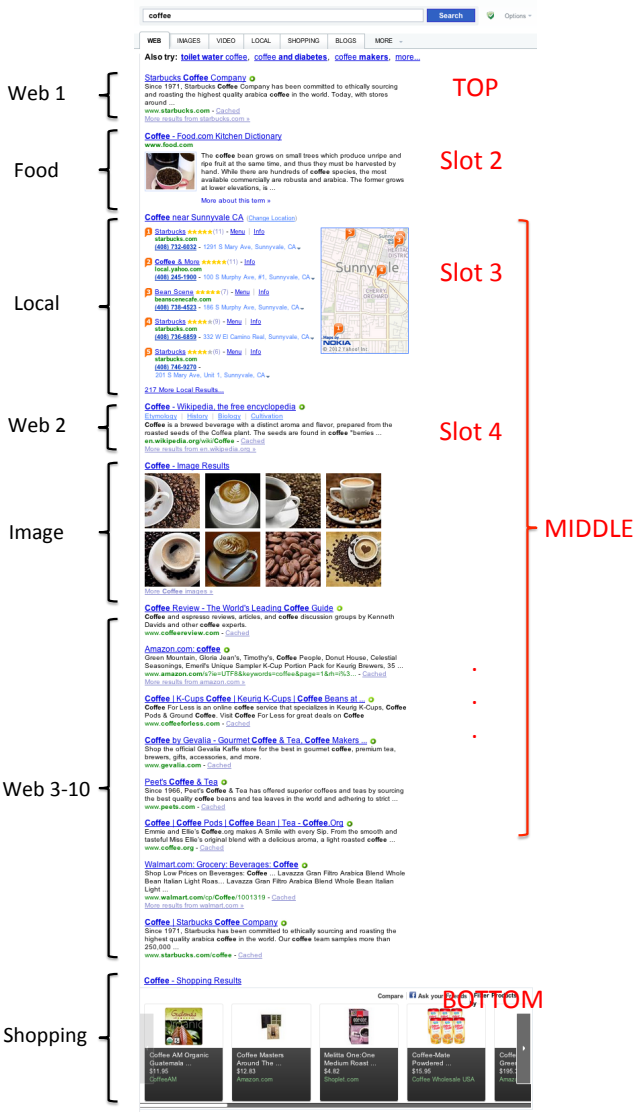


Figure 1: An example SERP for the query "coffee" shows image, food, local and shopping verticals at different positions.

position within the web text documents become particularly challenging. This problem is different from the core ranking problem in web search, where the textual web corpus of the same properties are compared with each other. In federated search, even though vertical databases usually have more structured representation, they are unlikely to share common features due to their heterogeneous nature. Hence we can not directly apply the same machinery used in web ranking.

Model-based triggering approach has become popular in recent years, and machine learning techniques have been used to fit the models. A new challenge that comes with this approach is how to gather enough data with labels to train the model. Hiring editors to judge the results can be expensive and is not scaleable. The judgements can be biased because many queries contain multiple intents. How does the system know if a user who searches for "starbucks" is looking for a nearby starbucks store or intends to navigate to Starbucks' office site? Furthermore, the intent of a query may also shift over time. In such cases, it is crucial to quickly identify the new intent. Another possible solution is by analyzing users click logs and obtaining training data for the model. Several studies have been conducted for the web documents ranking application, but few on the federated search setup. Different from the traditional web ranking problem, where all the documents are presented to the user in a ranked order, if a vertical has not been triggered given a query and/or has not been slotted above a web document, the system will receive no feedback, and thus will have no knowledge as to whether it matches the user's intent or if its results are preferred over a web document.

Several research projects have attempted to address related issues from different perspectives. In this paper, we present a unified search federation system. It addresses the challenge of aggregating the vertical results on the SERP in a unified and principled approach. The system currently serves three major markets of Yahoo! – USA[1], Taiwan[2] and Hongkong [3], and controls a dozen verticals in total. It is serving hundreds of millions of queries live each day, and has improved user metrics considerably. We formulated it as a contextual bandit problem, an approach which collects training data using an exploration/exploitation strategy, and updates its model based on the user-click feedback to maximize the total user satisfaction in the long run. We will review the related work in Section 2. Then, we formalize the definition of our problem, and introduce the framework of our federation system (Section 3). In Section 4, we present the method we implemented in each part of our system, and an offline evaluation method to evaluate our model. Finally, we show the performance improvement of our system over the old system on real web search traffic (Section 5).

2. RELATED WORK

The search federation problem has been studied in academia and the industry from various perspectives. Resource selection [4, 25] and query classification [22, 23, 15] have been studied extensively in information retrieval, and is very relevant to federated search. Most prior approaches to resource selection assume that different resources can be described using similar features [24], while in modern search engines verticals usually containing rich and a large variety of attributes such as images, videos and local listing. Hence, we are facing the problem that the vertical results can not be described using textual features, and we have to create a ranking function which can merge the results from these heterogeneous resources. Since several verticals are genre-specific, query classifier can potentially be used to trigger these verticals. However, even if query contains the intent, the vertical database may not have suitable content matching the query which will result in irrelevant result being triggered. Some work has been conducted on addressing the vertical selection problem directly. Li et al. [15] use lexical features and a query-click graph to propagate category labels to unlabeled queries for shopping and job verticals. Diaz [8] investigates the problem of whether or not to show the news vertical above the web results. Agruello et al. [2] use a machine

[1] http://search.yahoo.com
[2] http://tw.search.yahoo.com/
[3] http://hk.search.yahoo.com/

learning algorithm to learn models to decide the relevance of the vertical given a query, where human judges are used to generated ground truth labels for the verticals. Most of these papers focused on the problem of selecting the most relevant vertical and triggering them (on a fixed position), with the notable exception where Ponnuswami et al. [18] addressed the problem of placing one or multiple relevant verticals among the web documents. In this paper, the authors propose a machine learning framework which formulates the learning problem as a pairwise ranking problem and collects their training data through a randomized bucket. Our system, which has been developed independently from this paper, is very similar to it in the gist. However, Ponnuswami et al.'s system has several limitations. First, the triggering and slotting is based on threshold computed to match certain coverage target, while our system only slots a vertical in a certain position if it thinks the results are more relevant than the web documents below. Second, they do not have a principled and unbiased approach to evaluate their system offline. Last but not least, their system is not capable of continuously updating its model, which is critical for dynamic verticals like news.

Bandit approaches have recently become popular in web applications. Radlinski et al. use a bandit model to adapt retrieval results [20]. Li et al. model personalized recommendation of news articles as a contextual bandit problem and evaluate on offline data collected using random traffic [14]. Bandit approaches have also been applied to advertisement placement systems [16, 13, 10]. Although it seems natural to model federated search as a bandit problem, it is the first time this tool is used to formulate the problem.

Hiring human editors to generate labeled training data can be expensive, time consuming, and does not scale well. This motivates the efforts to generate labeled training data by analyzing the users' interaction with the SERP. Joachims [12] uses clicks and skips as preference judgments for learning ranking model. In [19], Radlinski and Joachims create preference labels from the engagement difference by randomizing consecutive search result pairs to users. Ji et al. [11] learn entities ranking model in a local vertical search engine from click-through logs. Motivated from the above works, we randomly slot verticals at available positions in the SERP and use the user's click-skip interaction for generating labeled train data. An interesting study conducted by Chen et al. [7] observes that user click behavior in federation search is very different from that in traditional web search, and calls for a new model to interpret the user's behavior. The reward definition in our framework is general, and can use click model and metrics as well.

3. PROBLEM STATEMENT AND FORMALIZATION

The problem statement for our federation system is to "trigger and slot verticals on SERP in a page-aware fashion such that the user satisfaction is maximized while subjecting to the business constraints".

In this section, we will first describe the system, then we will define the multi-armed bandit problem formally (Section 3.2) and show how federated search can be formalized as a bandit problem (Section 3.3).

3.1 Overview

When a user issues a query q in Yahoo!'s search engine, our system decides which verticals may be relevant to the query and passes the query to the corresponding vertical backends. Some of the backends may contain relevant content. These results are integrated together with the web documents, and composed into an unified SERP shown to the users by the system. The SERP is generated according to the principle of maximizing the expected user satisfaction while also subjecting to a few constraints. Constraints mainly come from the business restriction, such as the maximum number of verticals that can be slotted on a page, the positions a certain type of vertical can be shown at, not being allowed to change the ranking of web documents, certain coverage constraints for a vertical, etc.

3.2 A Multi-armed Contextual Bandit Formulation

The search federation problem can be modeled as a multi-armed contextual bandit problem. In the following section, we will briefly introduce the bandit problem. Due to space limitation, this is a very quick account of the bandit learning framework – interested readers are referred to [5][Section 6] for a comprehensive introduction.

A bandit algorithm A works in discrete rounds $t = 1, 2, 3, \ldots,$ T. At time t:

1. A observes a set of available arms or actions $\mathcal{A}$ and their associated contexts represented as feature vector $\boldsymbol{x}_{t,a}, \forall a \in \mathcal{A}$.
2. A estimates the expected rewards $r_{x,t}$ for each action based on the observed rewards from previous trails.
3. A chooses an action $a_t \in \mathcal{A}_t$ based on a defined action selection strategy, and the reward of the chosen action r_{t,a_t} is revealed to the algorithm.
4. A then improves its expected reward estimator and action selection strategy with the new observation $(r_{t,a_t}, a_t, \boldsymbol{x}_{t,a_t})$.

It is important to emphasize that no feedback is observed for other unchosen actions. The total reward of A is defined as $\sum_{t=1}^{T} r_{t,a_t}$, and the goal of the algorithm is to maximize the total rewards in the long run. If the underlying distributions of the rewards for each action given the context are known, then the problem becomes simple as in each round it can just *exploit* the action which receives the best expected reward. However, the true distributions of the rewards are not known, and the seemingly optimal action may in fact be biased and suboptimal due to imprecision in the algorithm's knowledge about the world. Because of the fact that the algorithm will not receive any feedback from the unchosen actions, in order to get a good knowledge of them, A has to gather information about them by choosing these seemingly suboptimal actions. This kind of strategy is referred as *exploration*. Exploration could harm the received reward in short term as some suboptimal action may be chosen. On the other hand, obtaining information about the action through exploration can help the algorithms get a better estimation of the actions' rewards, and in turn increase the long term overall reward. Obviously, a good tradeoff between exploration and exploitation is needed for the algorithm to work well.

3.3 Our Formulation

In the scenario of search federation, we may consider the SERP as an action. Under the constraint that the system cannot change the ranking of web documents, the action becomes putting available verticals at the allowed positions. In our case, verticals could be shown at the top of the page, between two web documents, or at the bottom of the page. Due to their different interfaces and business constraints, the allowed slot may be different for each vertical.

The user satisfaction given the SERP can be viewed as the reward. We need to first quantitatively define what is user satisfaction. Unfortunately, it is difficult to measure user satisfaction directly. Many definitions for the proxy of user satisfaction exist - all the way from simple click-through rates (CTRs), Long Dwell Time (LDT) clicks, to session based formulations such as path to success (PTS), search success, query reformulation etc. Even though session-level metrics can be reliable indicators of user satisfaction, they require handling of complex credit assignment problem, i.e. how much each of the slotting decisions eventually help towards user satisfaction, in order to be able to optimize for them. It is also possible to consider the full SERP configuration as an independent action. However, it will require a lot more data to construct the model due to the large number of possible actions (an enumeration of different combinations of the verticals on the allowed positions). Therefore, we chose to measure the reward of each vertical and web document, and the total reward of the action is the sum of the rewards of the items composing the SERP. In this paper, we decided to use click-skip based reward as the proxy for user satisfaction (as illustrated in Figure 2), as it outperformed click-only reward in our preliminary experiments. The click-skip reward assumes a cascade model of user behavior on SERP, i.e., a user starts examining the page from top to bottom and clicks only on the results that are perceived to be useful. The user stops when he/she finds the desired content. If no useful result is found, the search page is abandoned. When an item (vertical or web document) is clicked (one or more time), a reward of 1 is received; when it is skipped (no click on the item, and a click happens below the item), a reward of -1 is received. If none of these events occur for a result, it is assumed to be "abandoned" and gets a reward of 0. Such reward definition is far from perfect, but the notion helps overcome the position bias compared to metrics like CTR. For some verticals which show rich information through their template (e.g., a local vertical shows the telephone and address of the business, and a movie vertical has show times and rating listed), the users get their desired information without a click. Hence it is hard to distinguish "good" and "bad" abandonment. We chose to not use abandonments in training our reward prediction model. We will discuss the details of our reward predictor in the next section (Section 4.2). Other more sophisticated metrics could be taken into consideration, for example, LDT on a clicked vertical could be treated as a very positive feedback, while LDT on an abandoned page with content vertical (e.g., local) presented without a click and query reformulation could also be a positive feedback. We will investigate these opinions in our future study. In this paper, we will only use click-skip reward as our target metric.

Given a query, besides the textual features, we could also compute the intent of the query using query classifiers and extract various information from the backend response. More details about the features we are using are explained in Section 4.3.

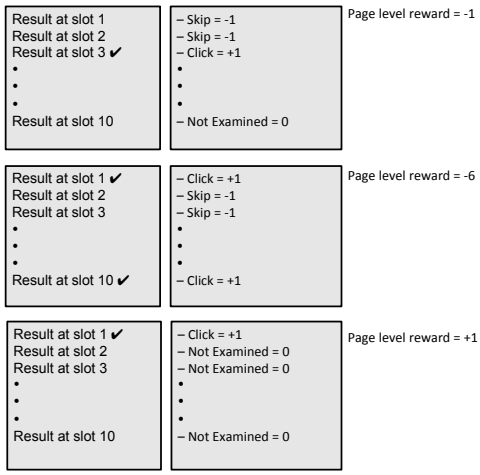


Figure 2: Figures illustrated examples how the click-skip based reward is computed. The left column shows the SERP pages with ten results and the results which receive a click are marked with a check mark. The middle column illustrates the rewards for each item on the pages.

4. OVERVIEW OF THE SYSTEM

Our system has runtime and offline components, Figure 3 illustrates the core components of the system. When a query comes to the search engine, it is sent to the federation layer to gather results from various backends. The layer decides whether to conduct exploration on the query or do exploitation. In either cases, the query is federated to all backends and from the content returned, features are computed. If exploration is chosen, the verticals are slotted randomly and the features along with rest of the context are logged. Otherwise, the model is used to score the results and then the slotting decisions are taken using the slotting mechanism. Offline, on the Hadoop grids, the logs are aggregated and processed to compute the rewards for each of the events of interest using the extracted features. Subsequently training and test data is generated from it and model(s) are trained and evaluated. The best model gets deployed to the runtime system. In the remaining part of this section we will explain each component individually.

4.1 Data Collection Through Exploration

One of the simplest and most straightforward bandit exploration algorithms is ϵ-greed [21]. In each round, with probability ϵ, the algorithm will choose a random action; and with probability $1 - \epsilon$, the algorithm will choose the action with the highest expected reward. When each query comes, the system decides if it should explore or exploit by tossing a biased coin. Exploration is done in a uniform random way with 100% dispatch rate to all the vertical backends, then the vertical(s) with content are randomly positioned in allowed slots with uniform probability. The randomly generated SERP is not influenced by the existing production ranking system. As a consequence, the model generated with this data is also devoid of any such influence from the

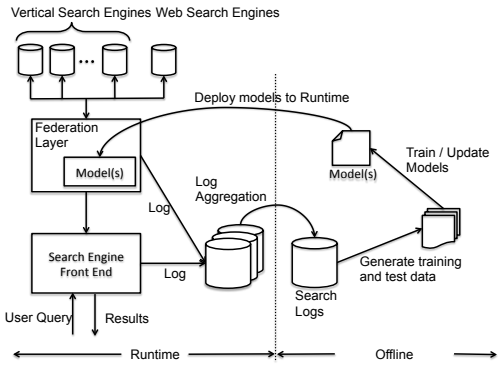


Figure 3: A high-level overview of our system. It can be divided into a runtime component and an offline component.

existing ranking system. Moreover, being randomly generated, the SERP is not subject to any positional bias as each vertical has a known probability to be placed in each of the available positions. This exploration SERP is then shown to the user who interacts with it in a normal fashion. The user interaction with the SERP, along with context information like query, user information, location, and features from backends, are recorded at runtime.

Offline, the system generates training and test examples from the recorded data. Each example corresponds to a slotting decision that was taken and consists of a 4-tuple (π, p, x, r), where π is the decision taken, i.e., which item (a vertical or a web document) is shown at a particular position on SERP, p, x and r are the probability the item gets slotted in the corresponding position, the feature vector and the reward for the decision, respectively. We use web_1, web_2, ..., web_{10} to denote the web documents from position 1 to position 10. Since we do not change the ranking of web document and they are always shown, their probabilities of getting slotted will always equal 1. The probability is then converted into an importance weight for the particular example using $\omega = 1/p$. The idea is that if an item has a lower probability to be shown at a certain position (due to business constraints and different vertical database coverage), then the example has to be treated more importantly than an item frequently (e.g., web documents) shown. Our learning algorithm has to take the importance weight of each example into consideration. As a result, the model is less likely to be affected by the noise from current system and setup. In practice, giving high importance weight to rare examples will also alleviate the data imbalance problem.

We use the whole production traffic to collect the data. Unlike other works (e.g., [18]) which continuously expose a small bucket of users to random results, each user will subject to random exploration, but with very low chance so that the user may not even notice it.

4.2 Statistical Model

We assume that the expected reward of an item π is linear in its d-dimensional context feature $\boldsymbol{x}_\pi$ with a unknown coefficient parameter $\boldsymbol{w}^*_\pi$, and is independent of its position and other results on the SERP. It can be represented as:

$$E[r_\pi|\boldsymbol{x}_\pi] = \boldsymbol{x}_\pi \cdot \boldsymbol{w}^*_\pi.$$

We use regularized linear regression with logistic loss to estimate the coefficient parameter from the training data collected through random exploration. To train the regressor, we use Vowpal Wabbit (VW) [1] as it has several desirable properties - it is very efficient at, both, the training and prediction phases, can scale to a large number of features and extremely big datasets, and can handle sparse features through feature hashing [26]. During training, we first initialize the model with a single pass of Stochastic Gradient Descent, then optimize the model using L-BFGS. In practice, we found batch optimization yields better results. VW is also capable of training the model in a distributed fashion. It allows us to retrain the model on the entire dataset with $10^8 - 10^9$ examples in a few minutes.

4.3 Feature Representation

A large part of our features come from the works of Diaz et al. [8, 2]. In addition, we also extract features from the results representing the quality of the backend contents. The big advantage of these features is that they allow us to filter out irrelevant or low quality content even if the query contains the vertical intent. The features used by our system can be summarized into the following categories:

- **Global result set features**: features derived from all the responses received. They indicate the content availability of each backend.
- **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 CTRs, user preferences, etc.
- **Backend features**: extracted from the backend responses. A list of statistical summary features such as relevance scores and ranking features of each 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 backend in the last few hours.

We have also tested using user information such as age group, gender and location, as features, but offline experiments show no gains from them. Perhaps more powerful personal features which can infer a user's interest could also be considered.

4.4 Slotting Mechanism

In exploitation mode, based on the predicted rewards of the participant verticals and other web documents, we use a locally greedy slotting algorithm to determine the final slotting decision. The algorithm starts from the top of the page, and decides the vertical to be slotted at each allowed position, while subjecting to the business constrains, from top to bottom. For each position, we pick a vertical from the remaining verticals with the highest expect reward, and compare it with the expected reward of the web document at the position below that position. For a vertical to be shown, its expected reward must be higher than the expected reward of that web document. For example, if a News vertical

is allowed to be shown at any position above the web_3 result, its expected reward must be at least higher than the expected reward of web_3, otherwise it will not be triggered. Our model maps these heterogenous entities like the web documents and verticals into an uniform reward target, and allows us to compare their relevance and rank them in a principled way.

4.5 Model Updates

Since users' information needs and vertical corpus change with time, it is desirable to periodically update the model to adapt to these changes. This is particularly true for News. Given a query, whether to trigger a News vertical changes with time. In our system, we update the model by retraining and redeploying the models periodically. Currently, the model update goes out daily, although the system allows us to update the model every five minutes.

4.6 Offline Policy Evaluation

Compared to a problem in the standard supervise setting, evaluation of the performance of a new system in federated search can be difficult because of the interactive nature of the task. The offline data is collected using a particular logging policy, and the action chosen by the logging policy may be different from the action chosen by the algorithm. Rewards are only observed if a vertical is triggered and slotted at a position examined by the user. Even though the same vertical is slotted at the same position, the user's behavior could be completely different if another vertical is shown on top of it. Due to the positional bias, a vertical slotted at a high position is likely to attract more clicks even though it is less relevant. Hiring human judges to do side by side tests is expensive, and does not scale well. It would seem that the only way to test the system is through A/B test on live traffic. However, it is not practical to deploy a model for online test until we are confident that the model is going to deliver a reasonably good performance. One solution is to build a simulator to evaluate the effectiveness of the algorithm. Motivated by the offline policy evaluator proposed by Li et al. [14] for the online recommendation systems, we propose a similar method for federated search. The method is simple to implement, and is unbiased, thanks to data collected through uniformly random exploration. Given a stream of events $(r_{t,a_t}, a_t, p(a_t), \boldsymbol{x}_t)$ collected through random exploration, where $a_t = \{\pi_t^{\text{TOP}}, \pi_t^{\text{SLOT2}}, \ldots, \pi_t^{\text{BOTTOM}}\}$ is the slotting decision for each position on the SERP and $p(a_t)$ is the probability for the SERP to be generated in uniform random slotting, the average reward for the T offline events can be computed as

$$\bar{r} = \frac{1}{T} \sum_{(r_{t,a_t}, a_t, p(a_t), \boldsymbol{x}_t)} \frac{r_{t,a_t} I(A(\boldsymbol{x}_t) == a_t)}{p(a_t)} ,$$

where I is the indicator function and $A(\boldsymbol{x}_t)$ is the slotting decisions made by our algorithm. Since we assume a cascade model of user behavior, for a page to be counted, the slotting decision at every position

$$A(\boldsymbol{x}_t) = \{\hat{\pi}_t^{\text{TOP}}, \hat{\pi}_t^{\text{SLOT2}}, \ldots, \hat{\pi}_t^{\text{BOTTOM}}\}$$

must match with the random exploration event, i.e., $\hat{\pi}_t^* == \pi_t^*$, otherwise the data will be discarded in offline evaluation.

5. RESULTS

Our system is a module of the Yahoo! search engine, and controls parts of its federation experience. The system has been deployed to three different markets – USA, Taiwan and Hongkong, and controls a dozen verticals in total. It serves hundreds of millions of queries live each day, and has improved user metrics like Click Through Rate (CTR), coverage and other user engagement metrics considerably. In this section, we start by describing our model training process and offline experimental setup. To show the effectiveness of our system, we then show the relative improvements over the old federation system from our bucket tests in USA and Taiwan. The old USA federation was built using editorial input, heuristic, and a hodge-podge of machine learning techniques applied in isolation to each vertical. For example, the triggering and slotting of the News vertical was decided by a whitelist generated by an offline model at some fixed positions, or several realtime models [9, 8], one for each position. The models were trained against CTR targets based on user feedback. On the other hand, the model for triggering and slotting of the Local vertical was trained against editorial targets. It classified the intent of the query, but did not consider the content match from the backend. The training data of the old Taiwan federation system [2] was generated by sending random queries against all the participating vertical backends, and the collected data was labeled by the editor. A GBDT [27] model was trained to scored the participating verticals. Similar to the method proposed in this paper, this system compared the scores of the vertical to web relevance scores at each of the allowed position and the winning result was shown there. The system requires the web ranking function to return a calibrated score comparable to the scores of the verticals.

We separate the SERP into three different regions: TOP, MIDDLE and BOTTOM (as shown in Figure 1), where any position between Web_1 and Web_{10} are categorized as the MIDDLE region. Our federation system has an agreement with various vertical owners to maintain a certain of level of coverage at TOP. This is a straight-forward process, during offline evaluation, if the coverage is lower than the agreed level, we artificially boost the score of the verticals to reach the desired coverage. The operation points can be obtained deterministically from the test data.

5.1 Data Collection and Model Training

Our data was gathered using a very small fraction of the search traffic for two weeks. We used the first 12 days of data for training, and the last 2 days as the validation set. The best modeling parameters were found through cross validation. The model which obtained the highest average reward in offline evaluation was used for online testing. The offline test results are reported in Table 1 and Table 2. In the test, we compare our results with two simple baselines. The first baseline is the click skip reward of the random exploration bucket. The second baseline is that the algorithm will always show a vertical at a fixed position whenever contents are available in the backend given a query. Although our offline evaluation policy works for the whole SERP, here we focused only on the TOP position in our offline evaluation. Because it will require much more test data in order to obtain a reliable estimation of the model's performance, due to the factor that the requirement for the exact matching between the exploration SERP and the offline SERP will

filter out a large part of the test data. In practice, we found that offline evaluation using only the TOP position provides a good guidance for how our model will perform online. As expected, our system outperforms the weak baselines significantly. When we always show a vertical at a fix position on the top, the CTR is relatively low because their content are irrelevant. One observation we had during data collection is that the percentage of the actual relevant contents is even less compared to the CTR numbers. Some users just click on the top results regardless of the relevancy due to positional bias. The "random" baseline performs relatively well because it also includes results from web_1, and the top web results are usually also very relevant thus they have high average CTR (due to recalls of different backends, the web results have a much higher chance to be chosen by the random baseline).

	Reward	CTR	Coverage
Our system	0.24	38.6%	–
News		29.6%	1.00%
Local		35.7%	1.99%
Shopping		19.9%	1.50%
Always News	-0.45	9.1%	28.17%
Always Local	-0.35	17.8%	13.04%
Always Shopping	-0.44	4.69%	26.26%
Random	-0.05	32.1%	–

Table 1: Offline test results on the random exploration data collected in USA in the span of two weeks.

	Reward	CTR	Coverage
Our system	0.78	48.00%	–
News		19.6%	1.20%
Blog		35.7%	1.49%
Knowledge		30.1%	1.49%
Shopping		31.1%	1.27%
Always News	-0.67	5.63%	22.94%
Always Blog	-0.43	16.09%	78.11%
Always Knowledge	-0.12	14.48%	78.62%
Always Shopping	-0.19	5.05%	78.63%
Random	-0.19	35.09%	–

Table 2: Offline test results on the random exploration data collected in Taiwan in the span of two weeks.

5.2 Online Evaluation

The online bucket test results are shown in Table 3 and Table 4. For the Taiwan results, we also report relative improvement on the overall CTR on all the verticals, relative improvement on ratio of LDT clicks on all the verticals, and relative improvement on average clicks on all the verticals per 1,000 visitors. A LDT click is defined as a click on the result where the time difference between the click and next event in the same timeout session is more than 100 seconds, or the click is the last event in the session. The ratio of LDT clicks is the percentage of LDT clicks in all the clicks. We argue that these metrics are a better proxy for user engagement and satisfaction compared to metric like CTR. We found that our system increases CTR and/or Coverage at different regions of the page, particularly the middle part of the page. For News, although the CTR has dropped, but the coverage has been increased significantly. Moreover, the Ratio of LDT Clicks and average clicks on all the verticals have been improved significantly compared to the old system in Taiwan. Notice that this metric is computed for all the verticals including the verticals which are not controlled by our system. If we only consider the participant verticals, the improvement will look more salient.

Vertical	Position	Change in CTR	Change in Coverage
News	TOP	-12.8%	81.846%
	MIDDLE	208.7%	98.6%
Local	TOP	10.9%	-25.13%
	MIDDLE	5.02%	98.6%
Shopping	TOP	31.3%	-39.2%
	BOTTOM	141.1%	84.0%

Table 3: USA bucket test results. The table reports the relative improvements over the old USA federation system. News results was collected from Oct 28 to Nov 30, 2011, while Local results were collected from Oct 03 to Oct 21, 2012.

5.3 Editorial Evaluation

For offline analysis, we requested editors to help us evaluate the relevance of the SERP page by conducting Side-by-Side test against the old USA production system. The editors are shown the SERP produced by two systems, and provide a grade {"New system much better", "New system better", "Same/can not judge", "Old system better", "Old system much better" } (the editors do not know which systems are new or old during the test). The judgements were influenced by two factors: relevancy and ranking. Here we focus on two verticals, News and Local.

In the News test, we sampled queries which triggered the News vertical by the new system and/or the old system, and took a snap shot of the SERP generated by the two systems immediately. The result is shown in Figure 4 (a). From the test, editors found our system has significantly increase the coverage compared to the old system, and can trigger trending News which are usually neglected by old system. In the Local test, we use a set of 1000 local intent queries collected in our previous study. The result is shown in Figure 4 (b). From the test, editors found that our system triggered Local DD for many queries with local intent which was not triggered by the old system. Most of the cases, where the old system performed "better" compared to the new system, are Local DD being slotted at a higher position compared to the old system. This is because the new system ranks the DD according to their relevancy compared to the web links. This difference would not affect the user experiences significantly.

5.4 Discussion

Our system leverages user feedback in a systematic manner and has positive impact on user satisfaction. It learns the user's intent using a data driven approach. Moreover, our system unifies the the decision making for all the participant verticals that allows to make content and context aware decisions that are optimal at the page level. For example, our model automatically learns that for queries like

Vertical	Position	Change in CTR	Change in Coverage
News	TOP	-13.04%	99.0%
	MIDDLE	26.6%	337.0%
Blog	TOP	27.92%	-39.78%
	MIDDLE	25.63%	221.14%
	BOTTOM	8.23%	-26.55%
Shopping	TOP	57.6%	58.6%
	MIDDLE	39.33%	232.97%
	BOTTOM	-1.83%	96.06%
Knowledge	BOTTOM	57.8%	-4.1%
Overall CTR on all the verticals		15.2%	
Ratio of LDT Clicks on all the verticals		18.9%	
Average Clicks on all the verticals per 1,000 visitor		25.3%	

Table 4: Taiwan bucket test results from July 7 through Jul 18, 2012. The table reports the relative improvements over the old Taiwan federation system.

"starbucks" and "apple store" majority of the users want to navigate to the official website, and place the Local vertical in the second position. Contention resolution between multiple relevant verticals is also data driven. It's hard to quantify how much coverage each vertical at different regions of the SERP should have, and historical data is rarely a good indication. We do not enforce a coverage target while training the model, and the system decides the triggering and slotting according to the maximization of user satisfaction principle. To honor the agreement with various vertical owners about coverage, we introduce an additional parameter for each vertical to allow us to tune the coverage at the TOP position. To figure out the optimal coverage, a prolonged series of bucket tests are needed. Ponnuswami et al. developed a method to characterize the performance of models using different operation points in one of their recent papers [17]. Similar methods could also be applied here to choose the operation points.

Exploration using a tiny percentage of online traffic allows us to collect training data for both the head and tail queries. Machine trained model using features from the backend and query classifiers also help the model generalize to queries which have never been seen before. Editors have found that our system performs much better compared to the old system trained using editorial data in terms of tail queries. For instance, queries like "home plumbing" and "dog coats" successfully trigger Local and Shopping verticals respectively. Although the exploration traffic is small, it may still affect user experience. On the other hand, random exploration allows us to get an unbiased estimation of the expected reward. Therefore, when adding a new vertical to the system, it is necessary to collect a sufficient amount of training data through exploration. After that, it is possible to further reduce the exploration traffic, and use user log from the production traffic to bootstrap the performance of the model. Another possible solution is to join the human judgement data with the noisy user feedback data for model training, as suggested in [6].

The system can be easily extended to include new verticals and can be deployed to new markets quickly. It is capable of updating its model every five minutes. Currently, we do not enable this function, and the model is updated daily with minimum human intervention. Through periodical updates, we have observed improvements for some dynamic verticals like News, while on verticals which are relatively stale the gain is negligible. To enable frequent automatic model updates, a robust monitoring process is needed to be setup first.

Due to the different characteristics of the language and backend in the three markets, we collected the training data separately and trained an independent model for each of the markets. When deploy to global markets with less traffic, it is also possible to train an unified model with training data collected from all the different markets if these markets have the same type of backends.

6. CONCLUSIONS AND FUTURE WORK

This paper describes a search federation system that makes effective use of implicit user feedback in a principled manner. We model the search federation system as a contextual bandit problem. The system works with heterogeneous backends and it learns the users preference from their interaction with the verticals. It has been successfully deployed in multiple markets and impacts millions of users.

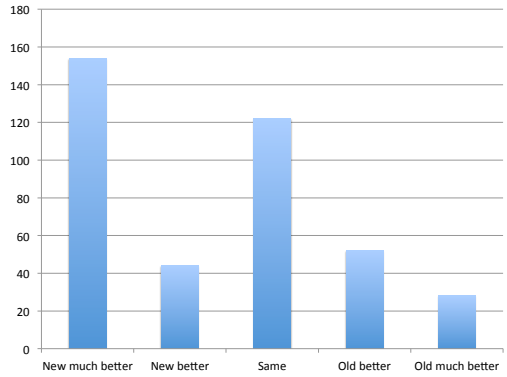
Future work will focus on developing a more sophisticated reward function which captures the user's satisfaction precisely. For example, the new reward function could integrate with other online metrics such as the LDT clicks, and consider the whole search session, as well as other browsing behaviors of the user. Another interesting direction is to extend the random based exploration decision to "guided" exploration where the system decides to explore only when it has low confidence about the incoming queries. Guided exploration can reduce the impact of exploration on user experience, and make efficient use of the limited traffic to collect user feedback on data where the system is mostly uncertain about. Various exploration approaches have been proposed in bandit literatures [3, 5, 14, 10], but little study has been conducted in ranking setup.

Acknowledgments

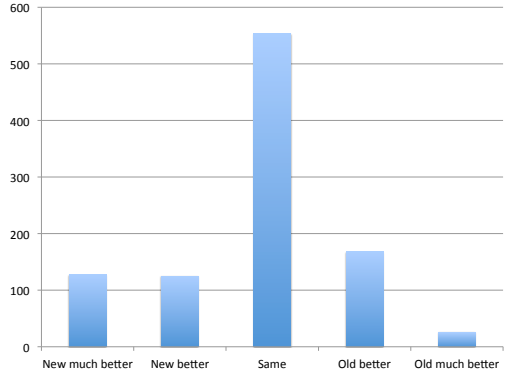
We are extremely grateful to our former colleagues Jeanfrancois Crespo, Olivier Chapelle, Fernando Diaz, Dumitru Erhan, John Langford and Sharath Rao who contributed to this work.

7. REFERENCES

[1] Vowpal wabbit. http://hunch.net/~vw/.



(a) News



(b) Local

Figure 4: Editorial evaluation for News (a) and Local (b).

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

[3] P. Auer. Using confidence bounds for exploitation-exploration trade-offs. *Journal of Machine Learning Research*, 3:397–422, 2002.

[4] J. Callan, Z. Lu, and W. B. Croft. Searching distributed collections with inference networks. In *Proc. SIGIR*, 1995.

[5] N. Cesa-Bianchi and G. Lugosi. *Prediction, learning, and games*. Cambridge University Press, 2006.

[6] O. Chapelle and Y. Zhang. A dynamic bayesian network click model for web search ranking. In *Proc. WWW*, 2009.

[7] D. Chen, W. Chen, H. Wang, Z. Chen, and Q. Yang. Beyond ten blue links: Enabling user click modeling in federated web search. In *Proc. WSDM*, 2012.

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

[9] F. Diaz and J. Arguello. Adaptation of offline vertical selection predictions in the presence of user feedback. In *Proc. SIGIR*, 2009.

[10] C. Gentile and F. Orabona. On multilabel classification and ranking with partial feedback. In *Proc. NIPS*, 2012.

[11] S. Ji, T. Moon, G. Dupret, C. Liao, and Z. Zheng. User behavior driven ranking without editorial judgements. In *Proc. CIKM*, 2010.

[12] T. Joachims. Optimizing search engines using clickthrough data. In *Proc. KDD*, 2002.

[13] R. Kleinberg, A. Slivkins, and E. Upfal. Multi-armed bandits in metric spaces. In *Proc. STOC*, 2008.

[14] L. Li, W. Chu, J. Langford, and R. E. Schapire. A contextual-bandit approach to personalized news article recommendation. In *Proc. WWW*, 2010.

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

[16] S. Pandey, D. Chakrabarti, and D. Agarwal. Multi-armed bandit problems with dependent arms. In *Proc. SDM*, 2007.

[17] A. K. Ponnuswami, K. Pattabiraman, D. Brand, and T. Kanungo. Model characterization curves for federated search using click-logs: predicting user engagement metrics for the span of feasible operating points. In *Proc. WWW*, 2011.

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

[19] F. Radlinski and T. Joachims. Minimally invasive randomization for collecting unbiased preferences from clickthrough logs. In *Proc. AAAI*, 2006.

[20] F. Radlinski, R. Kleinberg, and T. Joachims. Learning diverse rankings with multi-armed bandits. In *Proc. ICML*, 2008.

[21] H. Robbins. Some aspects of the sequential design of experiments. *Bulletin of the American Mathematical Society*, 58(5):527–535, 1952.

[22] D. Shen, R. Pan, J.-T. Sun, J. J. Pan, K. Wu, J. Yin, and Q. Yang. Q2c@ust: our winning solution to query classification in kddcup 2005. *SIGKDD Exploration Newsletter*, 7(2):100–110, 2005.

[23] D. Shen, J.-T. Sun, Q. Yang, and Z. Chen. Building bridges for web query classification. In *Proc. SIGIR*, 2006.

[24] M. Shokouhi and L. Si. Federated information retrieval. 2006.

[25] L. Si and J. Callan. Relevant document distribution estimation method for resource selection. In *Proc. SIGIR*, 2003.

[26] K. Weinberger, A. Dasgupta, J. Langford, A. Smola, and J. Attenberg. Feature hashing for large scale multitask learning. In *Proc. ICML*, 2009.

[27] Z. Zheng, H. Zha, T. Zhang, O. Chapelle, K. Chen, and G. Sun. A general boosting method and its application to learning ranking functions for web search. In *Proc. NIPS*, 2007.