Recommendation Projects at Yahoo!

Sihem Amer-Yahia,
Yahoo! Labs

1 Introduction

Recommendation is the task of providing content that is likely to interest users and enrich their online experience. At Yahoo!, the variety of user-facing applications and the nature and volume of data raise multiple recommendation opportunities and challenges. In this paper, we provide a brief description of several projects within Yahoo! Labs, in the context of Web Search (Section 2), interpreting users' clicks and browsing behavior (Section 3), and on the Social Web (Section 4).

2 In Web Search

2.1 Diversifying Presubmit Query Recommendations
Umut Ozertem, Emre Velipasasaoglu

Search assistance is an important feature in commercial search engines. Its two main flavors are *presubmit* and *postsubmit*. In presubmit, the aim is to auto-complete a user's partial query (referred to as prefix hereafter) to save time. In postsubmit, the goal is to suggest relevant follow-up queries to a user query. Presubmit poses a harder problem because (i) input from the user is a partially typed query and the user's intent is not defined, (ii) the algorithm needs to run at each character stroke, imposing a very strict computational complexity bound.

In Web search ranking, queries are associated with multiple intents, and when a user's intent is unknown, trading-off some relevance for result diversity is desirable [4, 6, 13, 32]. Presubmit is a clear case where intent is not defined. Common solutions suggest the most popular queries that match a user's prefix. However, mere frequency sorting leads to many cases with low utility (see Table 1). Therefore, in order to minimize effort (number of characters typed) before finding a useful suggestion, one needs to introduce diversity into the suggestion set.

The need for diversification has been studied in the context of query suggestion. There are two recent publications on this topic: one based on random walks over the query-URL bipartite graph [30], the other based on manifold ranking [38]. These methods define the diversification objective as "*given a query, generate a relevant but also diverse suggestion set*". Neither method is suitable for presubmit, where the query is yet to be defined. Hence, we adapt the problem to our context and define it as "*given a set of suggestions of size N, rerank the suggestions such that the overall utility in top K rank is maximized for all* $K \leq N$."

Diversity and relevance are conflicting objectives and optimizing both of them is known to be NP-hard. We develop a greedy algorithm that picks the query with highest marginal utility at each step [31]. We define a query utility as a function of the difference between its Web results and those of already presented query suggestions. For example the queries "facebook login" and "facebook home" are recognized as low utility queries when the

Bulletin of the IEEE Computer Society Technical Committee on Data Engineering

Table 1: Suggestions for prefixes **faceb** and **well** sorted by frequency and diversification

Frequency for **faceb**	Diversification for **faceb**		Frequency for **well**	Diversification for **well**
facebook	**faceb**ook		**well**s fargo	**well**s fargo
facebook login	farmville **faceb**ook		**well**s fargo bank	**well**sbutrin
facebook home	**faceb**ook farm town		**well**s fargo online	tom **well**ing
faceboklogin	**faceb**ook layouts		**well**sbutrin	**well**point
facebook friends	**faceb**ook ipo		tom **well**ing	dawn **well**s

query "facebook" precedes them. We define a measure of utility based on a user browsing model, and estimate the probability that each result can be examined in the search result page of the other query.

To build our model, we used 6 months of anonymized session logs of Yahoo! Search. We ignored queries that appeared less than 3 times. The model takes about 2.7M unique queries and 55.6M unique URLs in total, and builds a 2.7M $\times$ 2.7M query similarity matrix. We threshold this similarity matrix to construct a look-up table of duplicate queries. The look-up table consists of 1.56M queries with 20 duplicates each on average.

We designed an A/B test to measure effectiveness. The search engine traffic is split into two groups: the baseline (frequency sorting) and the control (greedy diversification algorithm) buckets. The user populations in both groups are uniformly sampled and their query volumes and distributions, i.e., the proportion of navigational and informational queries, are similar. If the greedy diversification algorithm changes the query suggestions we record this event as an *affected query*. Over affected queries, we found that the average character savings increased by 17%[1], and the time it takes the user to click on a destination URL decreases one second on average. A one-second time saving per search is considerable given the overall search volume. Over all the control bucket traffic, the increase in latency due to the diversification algorithm is only 1%.

2.2 Entity Ranking for Web Search

Changsung Kang, Srinivas Vadrevu, Ruiqiang Zhang, Roelof van Zwol, Lluis Garcia Pueyo, Nicolas Torzec, Jianzhang He, Yi Chang

A recent trend in Web search is to return structured entities. For example, a user looking for a movie can also see the movie plot, genre, cast, review and show times at her location while displaying related news articles, images, videos and tweets, wherever possible. The type of shown entities depends on the category of the query entity. For example, for movie queries, the goal is to show both lateral information like related movies and faceted information such as its cast.

In this work, the extraction of relevant entities relies on an Entity-Relationship knowledge base. The knowledge base contains an Entity-Relationship graph where nodes are entities and their attributes, and edges are semantic relationships among them. Our base consists of 4 million entities (100+ fine-grain entity types organized in a taxonomy) and 80 million relationship instances (hundreds of fine-grain relationship types, including both first order and derived relations). The main domains covered include Yahoo! Movies, Yahoo! TV, Yahoo! Music, Yahoo! Sports, Yahoo! OMG, and Yahoo! Travel. We have in the order of several hundreds facets for popular entities, and a few dozens for infamous ones.

We build a framework that computes entity features from various sources, including search logs, Flickr tags, and entity pairs detected in Twitter. From each source, we compute several probabilistic features that indicate the strength of the relationship between two entities based on their co-occurrence in various sources. Examples of features include atomic ones such as entity probability or entropy, symmetric features such as point-wise mutual information, asymmetric features such as conditional probability, and compound features that combine multiple ones. Two additional features are also computed from the graph: entity popularity based on a PageRank-like approach, and shared connections that indicate relationship strength in the graph.

[1] Average query length is 22, average prefix length for a suggestion click decreases from 13.5 to 12 after diversification.

We cast entity ranking as a supervised machine learning problem [37] with the goal of predicting relevance of the related entity to the query entity. We incorporate the categories of related entities into the loss function and leverage related entities from different categories to improve relevance.

Two types of experiments were conducted to validate our algorithms. The first set evaluates the performance of our approach on editorially judged entity pairs and the second aggregates user behavior on search results. Our data set contained 6000 query entities and 33000 entity-facet pairs. In the first experiment, editors provided a five-point relevance grade to indicate the match between a query entity and a facet entity. We showed that we achieve high Discounted Cumulative Gain (DCG). In the second experiment, we used pair-wise accuracy that reports how often users prefer a higher ranked entity over lower ranked entities in the related entity results. Comparison with several click preference models such as Cumulative Relevance [18], Skip CTR [22] showed the superiority of our Pair-wise Comparison Model. More details on this work can be found here [23].

3 Interpreting User Clicks and Browsing Behavior

3.1 Ad Recommendation using the Query-Ad Click Graph
Tasos Anastasakos

We address the sponsored search retrieval problem that is, the problem of finding and ranking relevant advertisements to a search query. A text ad on a search results page, otherwise known as sponsored link, is treated as a short document with a title, a short text and some keywords. We describe a technique to determine the relevance of an ad document to a search query using click-through data. The method builds on a collaborative filtering approach to discover and recommend new ads related to a query using a click graph. It is implemented on a graph with several million edges and easily scales to larger sizes easily.

Major search engines typically see traffic that contains millions of queries and a correspondingly large number of user-clicks on documents shown as results to those queries. While clicks are noisy due to click fraud, accidental or exploratory clicks, and position-bias, they can often be interpreted as a weak indicator of relevance. Several recent works have used click information to improve performance on various tasks. We propose an algorithm that operates on a large bipartite graph of queries and documents, where an edge between a query and a document is determined by whether users have clicked on the given document for the query. The weight on the edge represents the strength of the association measured as a function of several user metrics such as the number of clicks generated for the corresponding query and ad and the click-through-rate (CTR) of the query ad pair. We apply a collaborative filtering approach to first determine query-query similarity on this bipartite graph. Subsequently these query similarity scores are used to discover new documents that have not been shown for a given query before [7, 8].

The click-graph is built from a portion of Yahoo!'s sponsored search traffic over a 2-week period. It contains user queries and associated displayed and clicked ads. A typical graph consists of 27 million unique queries, 20 million unique ads and 51 million query-ad edges. The graph is regenerated every week in order to include recent queries and recent changes in user activity. We implemented a collaborative filtering algorithm on Map-Reduce and output a large lookup table of query-query and query-ad associations. (provide more brief details on algorithm or it will be hard to understand why you are doing better than baselines)

We evaluated the ad recommendations from our work off-line using human annotators/editors and online using live Web traffic to measure CTR improvements. We summarize our evaluation below.

Editorial Assessment : Query suggestions were labeled by trained editors who gave each ad a rating from 1 to 7, to reflect how strongly the ad meets the likely commercial intent or explicit need of the query. A rating of 1 is reserved for cases where only one perfect ad matches the query (e.g.,). Ratings 2 to 7 represent a decreasing degree of relevance.

Online Testing We ran our system live on a fraction (or "bucket") of Yahoo!'s traffic. The online traffic was divided so that a random subset of the user population (or a "bucket") receives candidates retrieved by the new algorithm. Our evaluation involved several millions users and collected statistics over a sufficiently long period of time (specify the time period). The goal of the experiment was to evaluate the monetization capability of the new algorithm using bucket testing measures [26].

We compared our algorithm to several baselines. We found 2-18% relative improvements in click rates, and 5-37% relative improvements in editorial judgments compared to the baseline.

Directions for future work include incorporating negative feedback and improving the performance for tail queries.

3.2 User Action Interpretation for Personalized Content Optimization
Anlei Dong, Jiang Bian, Srihari Reddy

Content optimization for recommender system is known as the problem of selecting content items to present to a user who is intent on browsing for information. Examples of content optimization are article publishing on portal Websites [2, 1], news personalization [16, 27], recommendation of dynamically changing items (updates, tweets, etc), and computational advertising [11, 33]. This work will address the variant that displays the best set of trending queries from the search engine in a module on the portal Website. This application is different from the task of query suggestion in Web search in the sense that it recommends popular queries to users from a certain pool of globally trending queries while query suggestion suggests queries relevant to those just submitted.

We are interested in user engagement when visiting a page. Engagement means the user examined or at least partly examined the content of a visited Web page. For example, when a user visits Yahoo! front page, it is possible she totally ignores the displayed *Trending Now* module content as she may be attracted to other modules such as the *Today* module, or goes to other services such as *Search* and *E-mail*. For a recommendation module, accurate Click-Through-Rate (CTR) estimation should be based on the events where users were really engaged, instead of all the events where the content of this module were displayed to users. We identify two categories of events regarding user engagement: *click event* and *non-click event*. In click events, it is obvious that the user likes the item she clicked, it also implies that other non-clicked items inside this module are not so interesting to the user compared to the clicked item. In contrast, non-click events are less informative because the user might not have examined the related module at all and the system cannot infer whether or not she is interested in the items belonging to the module. In this work, we only use click events as user feedback for online CTR estimation.

To validate our argument, we conduct experiments on real data from *Trending Now* on Yahoo! homepage. All data is anonymized in accordance with Yahoo!'s policy. We collected events in terms of *views* and *clicks* from a random learning bucket ([15]) during ten days from November 30th, 2010 to December 9th, 2010. This resulted in hundreds of millions of events with many millions unique users. Queries are randomly and displayed to users. An event records a user's action on the served queries on *Trending Now* ("*view*" is encoded as -1, "*click*" as 1). Specifically, we represent each event e as a set of tuples:

$$e = \langle u, t, p, q_p, a \rangle,\ p = 1, 2, \ldots, 10$$

where u denotes the user, t represents the time stamp for this event, q is the served query, p is the position at which q is displayed (there are totally ten positions on *Trending Now*), a is either *view* or *click*.

Users are assigned to groups according to a segmentation model. In this model, the system serves users in a group with models updated using feedback from users belonging to the group. User grouping is determined based on their demographics (e.g., age, gender) and other behavior and preference features [15]. To evaluate the recommendation model for *Trending Now*, we use the model to predict ranking scores for all candidate queries at a certain time stamp of each event, and rank them in descending order. For click events, we measure the ranking

position of the query that has actually been clicked by the user. More formally, for those clicks that actually happened at Position 1, we define $precision_i$ as the number of these clicks whose corresponding clicked items are ranked at Position up to i by the prediction of the model. This evaluation metrics has been proved to be unbiased towards online result [28].

We compare the modeling results using both all events and click events, where the model using all events is the baseline model. We observe from Table 2 that only using click events can significantly improve CTR estimation.

Table 2: Relative precision gain when training using click events over training on all events

Model	$prec_1$	$prec_2$	$prec_3$	$prec_4$	$prec_{10}$
Use click events	11.11%	7.05%	8.22%	7.70%	6.67%

Much research on user action interpretation has been conducted in the context of Web search. In particular, online user behavior modeling has been attracting much attention in recent years. Some work uncovered user behavior models based on controlled user studies [21, 35], while other studies focused on large-scale log analysis [36, 17]. Recently, some research [20, 19] has used eye-tracking studies to understand in detail how searchers examine search results, meanwhile dwell time interpretation has also attracted significant attention [25, 24] and has been extensively used for various information retrieval tasks [12, 29, 3]. However, user action interpretation has not been paid much attention in the studies of content optimization. Our work proposes to take deep analysis on user action interpretation in recommender system. In particular, we leverage user behavior information to sample training examples in order to remove those with little benefit for learning the model. To our best knowledge, none previous work has studied interpreting user actions in the context of content optimization.

4 On the Social Web

4.1 Recommending Travel Itineraries from Flickr Photos

Sihem Amer-Yahia, Nadav Golbandi, Ronny Lempel

Vacation planning is one of the frequent—but nonetheless laborious—tasks that people engage themselves with online; requiring skilled interaction with a multitude of resources. This paper constructs intra-city travel itineraries automatically by tapping a latent source reflecting geo-temporal breadcrumbs left by millions of tourists. For example, the popular rich media sharing site, Flickr, allows photos to be stamped by the time of when they were taken and be mapped to Points Of Interests (POIs) by geographical (i.e. latitude-longitude) and semantic (e.g., tags) metadata.

Leveraging this information, we construct itineraries following a two-step approach. Given a city, we first extract photo streams of individual users. Each photo stream provides estimates on where the user was, how long he stayed at each place, and what was the transit time between places. In the second step, we aggregate all user photo streams into a POI graph. Itineraries are then automatically constructed from the graph based on the popularity of the POIs and subject to the user's time and destination constraints [14].

We evaluated our approach by constructing itineraries for several major cities and comparing them, through a "crowd-sourcing" marketplace (Amazon Mechanical Turk), against itineraries constructed from popular bus tours that are professionally generated. Our extensive survey-based user studies over about 450 workers on AMT indicate that high quality itineraries can be automatically constructed from Flickr data.

This work is being integrated into Yahoo! Travel for suggesting intra-city itineraries of different time lengths. In particular, it is leveraged on mobile devices forinteractive itinerary planning as in [34].

4.2 Centralized and Distributed Query Personalization on Delicious
Sihem Amer-Yahia, Xiao Bai

Collaborative tagging systems represent huge mines of information. Yet, exploring these mines is challenging because of the unstructured nature of tagging and the lack of any fixed ontology. An appealing way to reduce the exploration space in collaborative tagging systems is to personalize the search by exploiting information from the social acquaintances of the seeker, typically users that exhibit similar tagging behaviors.

Given a user and a so-called user's network, composed of her social acquaintances, the relevance of an item to the user's query is a function of its popularity in that network, weighted by social distance. This challenge has been explored in both centralized and peer-to-peer environments. In [5], it was shown that scalable and efficient network-aware search must leverage user behavior to balance storage volume and response time. While search over information popular in a seeker's network can be achieved efficiently with straightforward adaptations of well-known algorithms, the storage volume of indexing every (seeker, keyword) pair, is daunting. Two space-saving solutions were explored: network clustering and behavior clustering.

In a centralized environment, it is expensive to maintain the inverted lists up-to-date in highly dynamic collaborative tagging systems where users frequently change their profiles by tagging new items. One possibility to circumvent this problem and improve the scalability is to personalize the search in a decentralized way. Besides being scalable and able to cope with dynamics, decentralized solutions inherently circumvent the danger of central authorities abusing the information at their disposal.

A natural, fully decentralized solution would consist for each user in a peer-to-peer system to locally store and maintain her network, enabling thereby efficient top-k query processing. The experiment on a 10,000-user *delicious* trace [9] confirms that the same result quality and query response time, as the most time efficient but storage consuming centralized approach, can be achieved with 0.011% of the total data stored at each user. Storage is no longer a severe issue in a decentralized setting. Yet, this requires each user to store all the profiles of her social acquaintances: these would then be massively replicated and hence hard to maintain. At the other extreme, a storage-effective strategy would consist for each user to store and maintain only her own profile and seek other profiles whenever a query is to be processed. Clearly, this optimizes the storage and maintenance issues but might induce a large number of messages and a large latency if profiles of acquaintances are to be consulted at query time. In addition, the profiles of temporarily disconnected users would be unavailable which, in turn, might significantly hamper the accuracy of the query processing.

A pragmatic solution [10], which we call P3Q (Peer-to-Peer Personalized Query processing), is a bimodal, gossip-based protocol to personalize the query processing in peer-to-peer systems. Users in P3Q periodically maintain their networks of social acquaintances by gossiping among each other and computing the proximity between tagging profiles. Each user maintains her network, namely a set of IDs of her social acquaintances. A user however only locally stores a limited subset of profiles, according to her storage capability. This removes on the one hand the bottleneck of a central server when the system keeps growing, and avoids on the other hand burdening any individual user in terms of storage. The maintenance of the network is performed in a lazy gossip mode, at a fairly low frequency to avoid overloading the network.

The querying scheme itself is based on an eager mode of the gossip protocol, i.e., with an increased frequency, and is biased towards social acquaintances. Each query is first computed locally by the querier, based on the set of stored profiles, providing an immediate partial result. Then the query, together with the list of profiles needed to compute it, is gossiped and computed collaboratively. The results are iteratively refined accordingly. Gossiping the query avoids saturating the network by contacting all the users in the personal network at the same time, and refreshes the part of the network originating from the querier, generating a specific wave of refreshments in the personalization process.

The analysis shows that the storage at each user only depends on her storage capability and does not increase with the system scale. The query processing time in gossip cycles can be approximated with $O(log_2 L)$, where L is the number of profiles in a user's network that contribute to the query processing but are not stored by

her. The experimental evaluation in a 10,000-user *delicious* trace confirms the scalability of P3Q: even if each user stores only 10 profiles out of thousands of social acquaintances in her network, corresponding to 12.5M bytes without any compression technique, the top-k queries can be accurately satisfied within 10 gossip cycles. Running the lazy mode every minute, even if all users simultaneously change their profiles, in half an hour, 95% of the stored profiles are updated, ensuring the freshness of the query results. Meanwhile, P3Q incurs acceptable overload in terms of bandwidth consumption: 13.4 Kbps are sufficient for maintaining the network and 91 Kbps are sufficient to compute a query. More adequate bandwidth, allowing both modes to run in higher frequency, would significantly decrease the network maintenance time as well as the query processing time.

References

[1] D. Agarwal, B.-C. Chen, and P. Elango. Fast online learning through offline initialization for time-sensitive recommendation. In *Proc. of KDD*, 2010.

[2] D. Agarwal, B.-C. Chen, P. Elango, N. Motgi, S.-T. Park, R. Ramakrishnan, S. Roy, and J. Zachariah. Online models for content optimization. In *NIPS*, 2008.

[3] E. Agichtein, E. Brill, and S. Dumais. Improving web search ranking by incorporating user behavior information. In *Proc. of SIGIR*, 2006.

[4] R. Agrawal, S. Gollapudi, A. Halverson, and S. Ieong. Diversifying search results. In *WSDM '09*, pages 5–14, New York, NY, USA, 2009. ACM.

[5] S. Amer-Yahia, M. Benedikt, L. V. S. Lakshmanan, and J. Stoyanovich. Efficient network aware search in collaborative tagging sites. *PVLDB*, 1(1):710–721, 2008.

[6] A. Anagnostopoulos, A. Z. Broder, and D. Carmel. Sampling search-engine results. In *WWW '05*, pages 245–256, New York, NY, USA, 2005. ACM.

[7] T. Anastasakos, D. Hillard, S. Kshetramade, and H. Raghavan. A collaborative filtering approach to ad recommendation using the query-ad click graph. In *CIKM '09: Proceeding of the 18th ACM Conference on Information and Knowledge Management*, pages 1927–1930, New York, NY, USA, 2009. ACM.

[8] T. Anastasakos, D. Hillard, S. Kshetramade, and H. Raghavan. A collaborative filtering approach to sponsored search. Technical Report YL-2009-006, Yahoo! Labs, Aug 2009.

[9] X. Bai, M. Bertier, R. Guerraoui, and A.-M. Kermarrec. Toward personalized peer-to-peer top-k pprocessing. In *Workshop on Social Network Systems (SNS)*, 2009.

[10] X. Bai, M. Bertier, R. Guerraoui, A.-M. Kermarrec, and V. Leroy. Gossiping personalized queries. In I. Manolescu, S. Spaccapietra, J. Teubner, M. Kitsuregawa, A. Léger, F. Naumann, A. Ailamaki, and F. Özcan, editors, *EDBT*, volume 426 of *ACM International Conference Proceeding Series*, pages 87–98. ACM, 2010.

[11] A. Broder. Computational advertising and recommender systems. In *Proc. of the 2nd ACM International Conference on Recommender Systems (RecSys)*, 2008.

[12] G. Buscher, L. van Elst, and A. Dengel. Segmentation-level display time as implicit feedback: a comparison to eye tracking. In *Proc. of SIGIR*, 2009.

[13] J. Carbonell and J. Goldstein. The use of mmr, diversity-based reranking for reordering documents and producing summaries. In *SIGIR '98*, pages 335–336, New York, NY, USA, 1998. ACM.

[14] M. D. Choudhury, M. Feldman, S. Amer-Yahia, N. Golbandi, R. Lempel, and C. Yu. Automatic construction of travel itineraries using social breadcrumbs. In *21st ACM Conference on Hypertext and Hypermedia (HT)*, 2010.

[15] W. Chu, S. T. Park, T. Beaupre, N. Motgi, and A. Phadke. A case study of behavior-driven conjoint analysis on yahoo! front page today module. In *Proc. of KDD*, 2009.

[16] A. Das, M. Datar, A. Garg, and S. Rajaram. Google news personalization: scalable online collaborative filtering. In *Proc. of the 16th International World Wide Web Conference (WWW)*, 2007.

[17] D. Downey, S. Dumais, D. Liebling, and E. Horvitz. Understanding the relationship between searchers' queries and information goals. In *Proc. of CIKM*, 2008.

[18] G. Dupret and C. Liao. Cumulated relevance: A model to estimate intrinsic document relevance from the clickthrough logs of a web search engine. In *Proceedings of the third International ACM Conference on Web Search and Data Mining (WSDM)*, 2010.

[19] L. A. Granka, T. Joachims, and G. Gay. Eye-tracking analysis of user behavior in www search. In *Proc. of SIGIR*, 2004.

[20] Z. Guan and E. Cutrell. An eye tracking study of the effect of target rank on web search. In *Proc. of CHI*, 2007.

[21] T. Joachims, L. Granka, B. Pan, H. Hembrooke, and G. Gay. Accurately interpreting clickthrough data as implicit feedback. In *Proc. of SIGIR*, 2005.

[22] T. Joachims, L. Granka, B. Pan, H. Hembrooke, F. Radlinski, and G. Gay. Evaluating the accuracy of implicit feedback from clicks and query reformulations in web search. *ACM Transactions on Information Systems (TOIS)*, 25(2), 2007.

[23] C. Kang, S. Vadrevu, R. Zhang, R. van Zwol, L. G. Pueyo, N. Torzec, J. He, and Y. Chang. Ranking related entities for web search queries. In S. Srinivasan, K. Ramamritham, A. Kumar, M. P. Ravindra, E. Bertino, and R. Kumar, editors, *WWW (Companion Volume)*, pages 67–68. ACM, 2011.

[24] D. Kelly and N. J. Belkin. Reading time, scrolling and interaction: exploring implicit sources of user preferences for relevance feedback. In *Proc. of SIGIR*, 2001.

[25] D. Kelly and N. J. Belkin. Display time as implicit feedback: understanding task effects. In *Proc. of SIGIR*, 2004.

[26] R. Kohavi, R. M. Henne, and D. Sommerfield. Practical guide to controlled experiments on the web: listen to your customers not to the hippo. In *KDD*, pages 959–967, 2007.

[27] L. Li, W. Chu, J. Langford, and R. E. Schapire. A contextual-bandit approach to personalized news article recommendation. In *Proc. of the 19th International World Wide Web Conference (WWW)*, 2010.

[28] L. Li, W. Chu, J. Langford, and X. Wang. Unbiased offline evaluation of contextual-bandit-based news article recommendation algorithms. In *Proc. of WSDM*, 2011.

[29] C. Liu, R. W. White, and S. Dumais. Understanding web browsing behaviors through weibull analysis of dwell time. In *Proc. of SIGIR*, 2010.

[30] H. Ma, M. R. Lyu, and I. King. Diversifying query suggestion results. In M. Fox and D. Poole, editors, *AAAI '10*. AAAI Press, 2010.

[31] G. L. Nemhauser, L. A. Wolsey, and M. L. Fisher. An analysis of approximations for maximizing submodular set functions. i. *Math. Programming*, 14(3):265–294, 1978.

[32] F. Radlinski and S. Dumais. Improving personalized web search using result diversification. In *SIGIR '06*, pages 691–692, New York, NY, USA, 2006. ACM.

[33] M. Richardson, E. Dominowska, and R. Ragno. Predicting clicks: estimating the click-through rate for new ads. In *Proc. of the 16th International World Wide Web Conference (WWW)*, 2007.

[34] S. B. Roy, G. Das, S. Amer-Yahia, and C. Yu. Interactive itinerary planning. In *ICDE*, 2011.

[35] J. Teevan, C. Alvarado, M. S. Ackerman, and D. R. Karger. The perfect search engine is not enough: a study of orienteering behavior in directed search. In *Proc. of CHI*, 2004.

[36] R. W. White and S. M. Drucker. Investigating behavioral variability in web search. In *Proc. of WWW*, 2007.

[37] 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 *Advances in Neural Information Processing Systems 20*, pages 1697–1704. MIT Press, 2008.

[38] X. Zhu, J. Guo, and X. Cheng. Recommending diverse and relevant queries with a manifold ranking based approach. In *SIGIR '10 Workshop on Query Representation and Understanding*, 2010.