Identifying “Best Bet” Web Search Results by Mining Past User Behavior

Eugene Agichtein
Microsoft Research
Redmond, WA, USA
eugeneag@microsoft.com

Zijian Zheng
Microsoft Corporation
Redmond, WA, USA
zijianz@microsoft.com

ABSTRACT

The top web search result is crucial for user satisfaction with the web search experience. We argue that the importance of the relevance at the top position necessitates special handling of the top web search result for some queries. We propose an effective approach of leveraging millions of past user interactions with a web search engine to automatically detect “best bet” top results preferred by majority of users. Interestingly, this problem can be more effectively addressed with classification than using state-of-the-art general ranking methods. Furthermore, we show that our general machine learning approach achieves precision comparable to a heavily tuned, domain-specific algorithm, with significantly higher coverage. Our experiments over millions of user interactions for thousands of queries demonstrate the effectiveness and robustness of our techniques.

Categories and Subject Descriptors

H.3.3 [**Information Search and Retrieval**]: Search process, Relevance feedback, Information filtering; H.3.3 [**Online Information Services**]: Commercial services, Web-based services

General Terms

Algorithms, Measurement, Design.

Keywords

Web search ranking, user behavior mining, web usage mining.

1. INTRODUCTION

Web search users are inexorably drawn to the top search result, often irrespective of the relevance of the document for the query. Furthermore, recent studies showed that users often do not examine lower-ranked results [12]. Hence, relevance at the top rank of the returned result list disproportionately affects user experience in web search.

The problem of selecting a good top result has been popularized by Google with the “I am feeling lucky!” button that automatically returns the top ranked document instead of a result list. We believe that for a large class of queries, we can *automatically* determine whether to propose such a “best bet” result, and what the result should be, by mining past user interactions with the search engine.

Specifically, we focus on the important class of queries usually referred to as “Navigational” [5]. These queries are submitted in order to get to a specific website without typing the actual URL. For these queries, the top result is often the destination a user is looking for. Clearly, many queries do not have such “best bet” results, and part of the problem is to identify the queries for which providing a specialized “best bet” result would help.

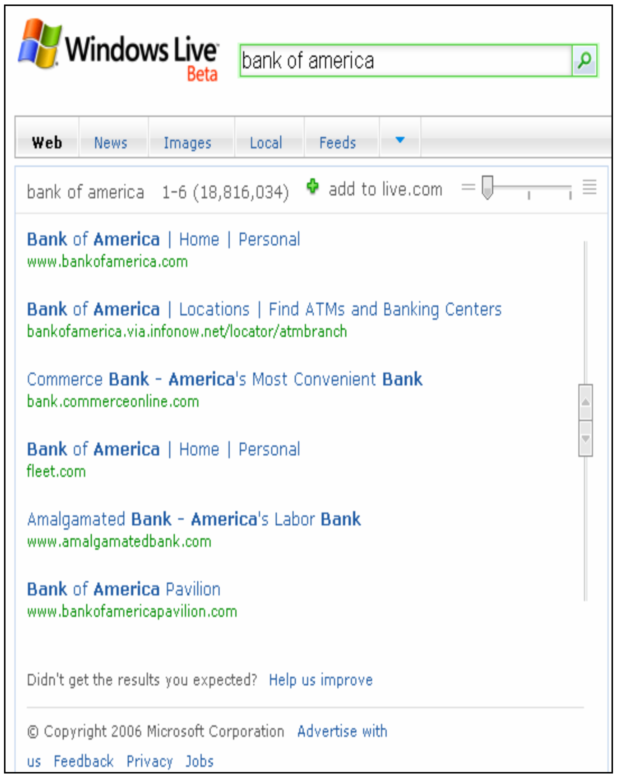
We consider different approaches for this problem. The natural and intuitive approach is to simply improve the ranker, e.g., by tuning the ranker to optimize accuracy of all top result. A related approach is to incorporate information from past user behavior to improve the overall ranking for previously submitted queries [1, 2]. A practical and effective variation of this general approach is to analyze past user behavior to propose “best bet” URLs using a set of rules created by a domain expert. Finally, and this is the approach we propose, we could automatically learn to detect “best bet” results and appropriate queries *simultaneously* by mining the past interactions with the search engine. Specifically, our contributions include:

- An investigation into general problem of representing user behavior such as to make it particularly amenable for suggesting the “best bet” search results (Section 3)
- Effective methods for incorporating user behavior into ranking and classification of search results (Section 4).
- Large-scale evaluation of the alternatives over millions of user interactions and thousands of queries (Section 6).

We review the related work in Section 7, and summarize our findings in Section 8, which concludes the paper.

2. THE IMPORTANCE OF TOP RESULTS

As we discussed, the top search result is disproportionably more important than the lower-ranked results. Consider a frequent query “bank of america” that is submitted to our search engine (Figure 2.1). This query is typically considered “Navigational” (i.e., a user wants to be directed to the appropriate website). Such queries account for nearly 30% of unique queries submitted [5, 19]), and an even larger fraction of queries if we consider the frequency of each query. As Figure 2.2 shows, the overwhelming majority of users prefer the top result (The official Bank of America website), which they might use to perform transactions or navigate to specific areas of the site. In this case, the ranking is perfect, in the sense that the top result satisfies the search need for more than 90% of the users. This example suggests that while it is important to return relevant results in general, returning a correct result in the top position for navigational queries can significantly improve the user experience.



bank of america 1-6 (18,816,034)

Bank of America | Home | Personal
www.bankofamerica.com

Bank of America | Locations | Find ATMs and Banking Centers
bankofamerica.via.infonow.net/locator/atmbranch

Commerce Bank - America's Most Convenient Bank
bank.commerceonline.com

Bank of America | Home | Personal
fleet.com

Amalgamated Bank - America's Labor Bank
www.amalgamatedbank.com

Bank of America Pavilion
www.bankofamericapavilion.com

Didn't get the results you expected? Help us improve

© Copyright 2006 Microsoft Corporation Advertise with us Feedback Privacy Jobs

Figure 2.1: Sample set of results for query "bank of America".

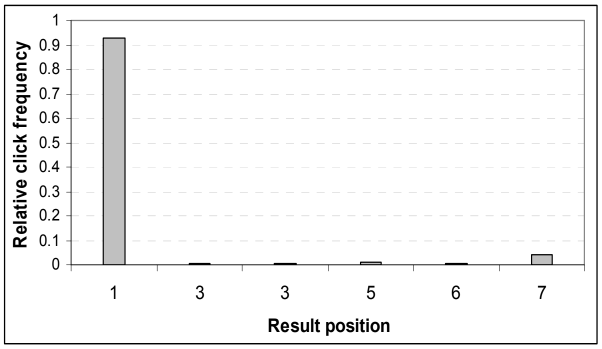


Figure 2.2: Click frequency distribution for query "bank of america", with the top result (www.bankofamerica.com) receiving more than 90% of all clicks.

Furthermore, the top result in web search has another important property: users have a strong bias towards clicking on the top result. Figure 2.3 shows the relative clickthrough frequency for more than 120,000 searches performed for 3,500 queries randomly sampled from query logs over a three week period. The queries were sampled across all query frequency ranges, to include a representative sample of both rare and frequent queries. The aggregated click frequency at result position *p* is calculated by first computing the frequency of a click at *p* for each query (i.e., approximating the probability that a randomly chosen click for that query would land on position *p*). These frequencies are then averaged across queries and normalized so that relative frequency of a click at the top position is 1. The resulting distribution agrees with previous observations about the bias in clicking top ranked results – users click more often on results that are ranked higher. Furthermore, there is a large bias towards the top result – the click frequency is nearly twice higher than on the next (second) result.

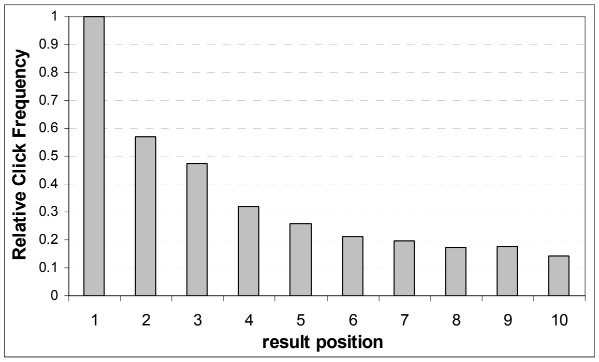


Figure 2.3: Relative click frequency for top 10 result positions over 3,500 queries and 120,000 searches.

This suggests that users may have such a strong bias towards the top result that they may click on the top result even if it is not relevant. Figure 2.4 reports the distribution of clicks on the relevant and non-relevant documents for these queries, which will motivate our emphasis on getting the top result correctly. Specifically, we report the aggregated click distribution for queries with varying Position of Top Relevant document (PTR). For example, for queries with top relevant result in position 3, the relative click frequency on the *non-relevant* results at position 1 is *higher* than click frequency on the *relevant* results in position 3. This indicates that users often click on the top result without considering the relevant results ranked below the top position.

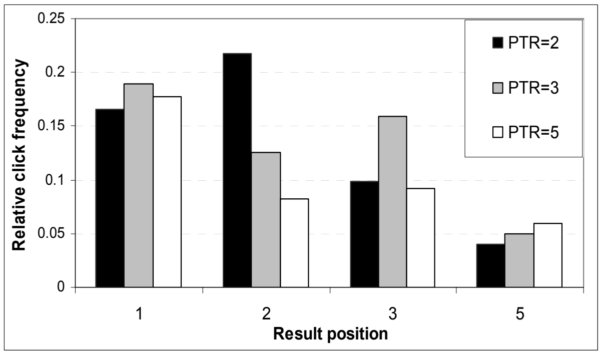


Figure 2.4: Relative click frequency for queries with varying PTR (Position of Top Relevant document).

For these reasons of impact on the web search experience, we specifically focus on the problem of returning an excellent "best bet" result in the top position when appropriate – i.e., when an overwhelming majority of users had exactly this result in mind when issuing a query. Specifically, our goal is to identify pairs of queries and corresponding results <*q,u*> such that past user interactions with the query *q* clearly indicate that users overwhelmingly prefer the result *u* over all other results for this query. Note that this problem is different from generic ranking in the sense that we do *not* propose a "best bet" result for every query, but rather for only the queries for which we can confidently predict a likely correct candidate. Furthermore, not all queries have one "best bet" result that could be returned. For example, a query "data mining research" does not have a best bet result, as there may be multiple relevant results and sub-areas of interest.

It is often difficult to directly understand the user intent. However, we can observe, and mine, the past *behavior* of users interacting with the search results as a proxy for understanding the user intent and preferences for the results. In the next section we describe how we represent the user behavior, to be used by machine learning-based and domain specific algorithms introduced in Section 4.

3. MINING USER BEHAVIOR

Although user implicit feedback (e.g., clicks) contains much noise, we believe it also contains valuable information on what users like. Our goal is to accurately interpret the *noisy* user feedback obtained by tracing user interactions with the search engine. Interpreting implicit feedback in real web search setting is not an easy task. We characterize this problem in detail in [1,2], where we motivate and evaluate a wide variety of models of implicit user activities. The general approach is to represent user actions for each search result as a vector of features, and then train a ranker on these features to discover behavior patterns indicative of relevant (and non-relevant) search results. We now briefly summarize our features and model, and the machine learning approach of interpreting user feedback (Section 3.2).

3.1 Representing User Actions as Features

We model observed web search behaviors as a combination of a "background" component (i.e., query- and relevance-independent noise in user behavior, including positional biases), and a "relevance" component (i.e., query-specific behavior indicative of the relevance of a result to a query). We design our features to take advantage of aggregated user behavior. The feature set is comprised of *directly observed* features (computed directly from observations for each query), as well as query-specific *derived* features, computed as the deviation from the overall query-independent distribution of values for the corresponding directly observed feature values.

A sample of features that we use to represent user interactions with web search results is listed in Table 3.1. This information was obtained via selective instrumentation from queries submitted to our search engine. We include the traditional implicit feedback features such as clickthrough counts for the results, as well as our novel derived features such as the deviation of the observed clickthrough number for a given query-URL pair from the expected number of clicks on a result in the given position.

Presentation Features	
ResultPosition	Position of the URL in Current ranking
QueryTitleOverlap	Fraction of query terms appearing in result Title
QueryURLOverlap	Fraction of query terms appearing in the URL
QuerySummaryOverlap	Fraction of query terms in result summary
Clickthrough and Browsing features	
TimeToClick	Seconds between query and click on the URL
TimeToFirstClick	Seconds between query and click on any result
ClickFrequency	Fraction of clicks for this query on this URL
ClickDeviation	Deviation from expected click frequency
IsNextClicked	1 if there is a click below, 0 otherwise
IsPreviousClicked	1 if there is a click above, 0 otherwise
IsClickAbove	1 if there is a click above, 0 otherwise
IsClickBelow	1 if there is click below, 0 otherwise
TimeOnPage	Result page dwell time
CumulativeTimeOnPage	Cumulative time for all subsequent pages visited

Table 3.1: A sample of features used to represent user behavior history for each query and result URL pair.

The features listed in Table 3.1 are not all the features used, but give a good idea of the variety and the type of user behavior that can significantly enrich the ranking model. We now briefly overview our general approach for deriving a user behavior model. We describe the actual methods in detail in Section 4.

3.2 Deriving a User Behavior Model

To learn to interpret the observed user behavior, we correlate user actions (e.g., the features in Table 3.1) with the explicit user judgments for a set of training queries. We find all the instances in our session logs where these queries were submitted to the search engine, and aggregate the user behavior features for all search sessions involving these queries.

Each observed query-URL pair is represented by the superset of features in Table 3.1, with values averaged over all search sessions, and assigned one of the two labels, *1* or "best bet" and *0* or "not best bet", as assigned by explicit relevance judgments. These labeled feature vectors are used as input to the machine learning and domain-specific algorithms, described next.

4. DISCOVERING BEHAVIOR PATTERNS

Describing user behavior as a set of *features* in the user behavior "space" has an important advantage: instead of writing domain-specific algorithms to interpret behavior, we can use general machine learning methods that can be *trained* given labeled examples of the behavior feature vectors for a given result, and the target relevance label for the result. We first describe a general machine learning framework for automatically tuning a web search ranking function given labeled examples for a given feature set. We then briefly describe how the behavior features can be incorporated into ranking directly (Section 4.2). We then describe a contrasting domain-specific approach developed for selecting the best top result by interpreting past user interactions (Section 4.3). Finally, we introduce our general machine learning approach for the top-result problem that treats it as a binary classification problem (Section 4.4). The methods described in this section will be empirically evaluated in Sections 5 and 6.

4.1 Learning to Rank Web Search Results

A key aspect of our approach is exploiting recent advances in machine learning, namely trainable ranking algorithms for web search and information retrieval (e.g., [6, 11]). In our setting, explicit human relevance judgments (labels) are available for a set of web search queries and results. Thus, an attractive choice to use a supervised machine learning technique is to learn a ranking function that best predicts relevance judgments.

RankNet [6] is one such algorithm. It is a neural net tuning algorithm that optimizes feature weights to best match explicitly provided pairwise user preferences. RankNet uses a probabilistic cost function that is robust to some noise in the labeling. While the specific training algorithms used by RankNet are beyond the scope of this paper, it is described in detail in [6] and includes extensive evaluation and comparison with other ranking methods. An attractive feature of RankNet is both train- and run-time efficiency – runtime ranking can be quickly computed and can scale to the web, and training can be done over thousands of queries and associated judged results.

We use a 2-layer implementation of RankNet in order to model non-linear relationships between features. Furthermore, RankNet can learn with many (differentiable) cost functions, and therefore can automatically learn a ranking function from human-provided labels, an attractive alternative to heuristic feature combination techniques. Hence, we will also use RankNet as a generic ranker over the user behavior features described above.

4.2 Ranking with Behavior Features

Modern web search engines rank results based on a large number of features, including content-based features (i.e., how closely a query matches the text or title or anchor text of the document), and query-independent page quality features (e.g., PageRank of the document or the domain). In most cases, automatic (or semi-automatic) methods are developed for tuning the specific ranking function that combines these feature values.

Hence, a natural approach is to incorporate implicit feedback features directly as features for the ranking algorithm. During training or tuning, the ranker can be tuned as before but with additional features. At runtime, the search engine would fetch the implicit feedback features associated with each query-result URL pair. This method is described in detail in [2], and has been shown to significantly improve the search result relevance. This model requires a ranking algorithm to be robust to missing values: a large fraction of web search queries are unique, with no previous implicit feedback available.

However, for the frequent, navigational queries that we target in this paper, past user interactions are often available, suggesting using intuitive rules defined over the behavior features to identify the most promising "best bet" results.

4.3 Domain-Expert Analysis of User Behavior

We can have search domain experts analyze user query logs including user behavior information, and develop ad hoc algorithms and rules to create best bets. In this paper, we refer this approach to DomainAlgorithms.

Depending on the quality and quantity of user logs processed, the experience of the experts, as well as tools available, the quality of the best bet results generated by DomainAlgorithms will vary. This approach works in practice, and, as we will show empirically, exhibits high accuracy. One advantage of DomainAlgorithms is that it does not rely on labeled training examples. Nevertheless, as the domain expert needs to analyze the logs and implement the algorithms and evaluate the accuracy, the resulting algorithms require significant effort to develop. As we only use DomainAlgorithms for comparison purposes, we do not describe the details of the rules constructed and incorporated into the system.

4.4 Building a Classifier to Detect "Best Bets"

While the algorithms above are likely to have high precision, they would have to be re-tuned and expanded as user behavior patterns change, as general ranking improves, and as spammers figure out ways to attack the rankings. In contrast, a more general *trainable* data mining approach is likely to be more robust and will discover regions in user behavior space that may not be easily engineered or even detected by manual analysis and algorithm engineering.

We model the problem as binary classification, i.e., learning to partition the user behavior space into regions where a result looks like a "best bet" or not in the past user behavior. The classifier is trained on user behavior feature vectors (Section 3.1) each with associated label derived from explicit relevance ratings. A classifier is then trained to partition the behavior feature space into regions where a new result is likely to be a "best bet" because users interact with the result similarly to known "best bet" results (i.e., map to close positions in the user behavior space).

We experimented with different classifiers (e.g., SVM implementations) and settled on a decision tree classifier, as the most intuitive and easily interpretable and also with highest accuracy on the development set. Additionally, a decision tree model also makes it easy to extract rules that could be examined and visualized to develop insights into user behavior. Specifically, we use the WinMine implementation of deriving decision trees from Bayesian networks, developed by Chickering [7] and publicly available for download[1].

5. EXPERIMENTAL SETUP

The experiments we report were all done on real data and with real user feedback datasets. The reranking experiments were performed *off-line* – i.e., by comparing the output of the systems with previously labeled relevance judgments for the search results. The methods compared are described next in Section 5.1 and are evaluated on the datasets in Section 5.2 using metrics in Section 5.3. These settings will be used for reporting the experimental results in Section 6.

5.1 Methods Compared

As we discussed the possible approaches to "best bet" result generation include web search ranking over document and query features, ranking that incorporates user feedback, and two methods specifically designed for selecting top search results:

- **RankNet**: A state of the art web search ranking system trained to order query results using features such as term match, match positions, web topology features, and hundreds of other features describing the similarity of a query to the candidate search result document (Section 4.1)

[1] http://research.microsoft.com/~dmax/WinMine/Tooldoc.htm

- **RankNetExtended**: The same RankNet system as above, but with a richer feature set that incorporates the past user behavior features into the ranking process (Section 4.2, reference [2])
- **DomainAlgorithms:** Algorithms or rules developed based on domain expertise and selective query log and user interaction analysis (Section 4.3)
- **BehaviorClassifier:** The general classification-based method trained on examples of "positive" and "negative" user behavior feature vectors for automatically discovering patterns in behavior that indicate a "best bet" result (Section 4.4)

The methods described above span the range from algorithms specifically designed for the "best bet" problem to general ranking and classification algorithms that were automatically tuned for the task. As we will show, the classification approach can rival the heavily engineered algorithms in accuracy, while also capturing cases not covered by manually developed algorithms.

5.2 Datasets

An important contribution of our work is applying implicit feedback techniques to important problems with real, noisy data collected "in the wild" by real users interacting with a web search engine. The actual ranking and evaluation experiments were done off-line, i.e., using a static set of queries, results, and explicit relevance judgments for the query results.

The judgments were collected by asking annotators to associate a relevance label for each result retrieved in response to a randomly selected set of queries from the search engine query logs. The queries were sampled by token without replacement, i.e., frequent queries have a high chance to be included, but the sample also contains a large number of "tail", or less-frequent queries.

For these experiments, we used 7,000 labeled queries that also had at least one of the results clicked (i.e., had a minimum of user interactions recorded). The user behavior data was collected over the period of 8 weeks, which contained the total of 1.2 million search requests for the queries considered, and the total of nearly 10 million user interactions with results of these search requests. The queries and associated feature sets were split three-fold for 2/3 train/test split, and a three-fold cross validation was conducted. The training and test sets were disjoint, randomly split by query. On average the training set contained 4,600 queries and 12,800 labeled URLs, and the test set contained 2,300 queries and 6,400 URLs with explicitly judged relevance labels and were clicked on at least once.

5.3 Evaluation Methodology and Metrics

The "best bet" result setting as introduced in Section 2 is essentially a standard information retrieval problem, except that we allow the system to avoid making a bet for "hard" queries, in which case the overall system backs off to the generic ranking of the results. Therefore, we use standard information retrieval metrics of Precision at 1 and Recall at 1 (see Salton et al. [20]), defined below, to compare the approaches.

Precision: the fraction of the queries for which a system proposed a "best bet result", where the proposed result was judged to be a "best bet" for the respective query.

Recall: the fraction of the queries with at least one "best bet" result that were successfully returned by a system.

The precision and recall metrics can also be combined into a single number (e.g., F-measure), but for this evaluation we felt it was important to explicitly consider the trade offs made between optimizing the recall and precision of a "best bet" system.

6. RESULTS

We now experimentally compare the methods described in Section 5.1 over the datasets and metrics of the previous section. The main results are summarized in Table 6.1.

The generic ranking methods are substantially outperformed by the methods specifically designed for the "best bet" problem. The precision of RankNet on document and query features is 0.239. Incorporating user behavior features into ranking boosts precision by 10% absolute to 0.331 (for 38.5% precision gain). However, a classifier specifically trained to recognize user behavior for "best bet" results achieves precision of 0.753 (at slightly lower recall), for precision gain of over 216%**.** Interestingly, the general classification-based method performs nearly as well as the heavily tuned set of domain-specific algorithms, but with much higher recall (i.e., coverage): BehaviorClassifier recognizes almost 30% of the "best bet" results, while DomainAlgorithms rules are able to detect less than 20% of the queries with "best bet" results.

Method	*Precision*	*Recall*	*Precision Gain (%)*
RankNet	0.239	0.239	-
RankNetExtended	0.331	0.331	38.5%
BehaviorClassifier	0.753	**0.299**	216%
DomainAlgorithms	**0.758**	*0.185*	**218%**

Table 6.1: Precision and Recall of the top results for RankNet, RankNetExtended, BehaviorClassifier, and DomainAlgori-thms methods over 2,300 test queries.

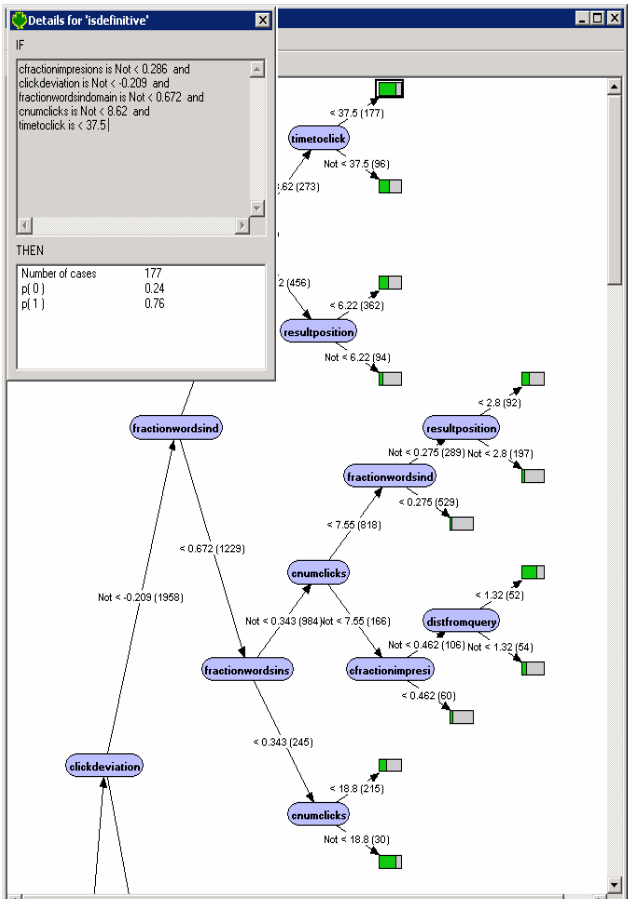
An important reason for the difference in precision exhibited by the ranking methods is that a ranker must *always* propose a URL for every query, whereas the BehaviorClassifier and DomainAlgorithms only propose a result when past user behavior indicates a promising candidate. Furthermore, the evaluation is in a sense different from standard in that a reasonable, but not perfect result would still be considered success in general ranking evaluation, whereas in this evaluation we only consider a "best bet" result (i.e., one that is likely to satisfy completely the majority of users) to be relevant.

In summary, our results show a drastic improvement in precision for the specialized "best bet" methods over general ranking approaches, and also show that a general classification approach can rival and outperform heavily-tuned domain-specific algorithms with proper feature representation and training.

6.1 Discussion

Before we discuss the implications of our results, we first present some intuitions into what the behavior models automatically discovered by classifying past user behavior. A decision tree fragment and a corresponding rule for one of the leaf nodes is shown in Figure 6.1. In our implementation, the label of "1" indicates a "best bet" search result, and label of "0" indicates a less relevant result. Hence, the leaves of the tree with high probability of the "1" label are the regions of the behavior space most likely to contain "best bets".

The splits in the decision tree, as translated into rules, are often intuitive in retrospect, but would have been difficult to design in advance. For example, the sample rule in Figure 6.1 states that a “best bet” result is likely with probability of 0.76 if there is large overlap of query words and with the domain of the result, there is a large number of clicks on the result, and the users take less than 37 seconds on average to click on the result.



Details for 'isdefinitive'

IF

```
cfractionimpresions is Not < 0.286  and
clickdeviation is Not < -0.209  and
fractionwordsindomain is Not < 0.672  and
cnumclicks is Not < 8.62  and
timetoclick is < 37.5
```

THEN

Number of cases	177
p(0)	0.24
p(1)	0.76

Figure 6.1: Example rule (leaf) of decision tree for selecting a top result based on past user behavior patterns.

6.2 Practical Impact

While the recall of returning an excellent “best bet” result for less than 30% of unique queries may seem small, in terms of user experience the effect is more significant. The queries for which there is sufficient amount of user interaction to make a good bet are also more frequent – in other words, in terms of query volume, these queries cover far more than 30%. Hence, a significant improvement of the top 1 result of the magnitude described noticeably improves the actual web search user experience.

Furthermore, the overlap of the “best bets” generated via classification with the general ranking methods is small. Hence, the improvement seen in isolation is expected to persist in the “live” setting where a classifier-based system might back-off to the ranking method if there is no top result to propose with high confidence based on past user behavior.

Using a general classification method also has important benefits. As users behavior evolves, and as click fraud and spam operators become more sophisticated, it would be difficult to maintain and update the rules and the code in the domain-specific algorithms. In contrast, our classification approach is easily amenable to updates (i.e., can be retrained over time and with new user behavior data over existing relevance judgments). Hence, the impact of our approach goes beyond the specific relevant improvements (which are substantial). In summary, we presented and empirically validated a flexible and effective approach to mining past user behavior data to suggest “best bet” results for appropriate queries.

7. RELATED WORK

Ranking search results is a fundamental problem in information retrieval and web search. Most common approaches primarily focus on similarity of query and a page, as well as the overall page quality [3, 20]. However, with increasing popularity of search engines, implicit feedback (i.e., the actions users take when interacting with the search engine) can be used to improve the rankings. Implicit relevance measures have been studied by several research groups. An overview of implicit measures is compiled in Kelly and Teevan [13]. This research, while developing valuable insights into implicit relevance measures, was not applied to improve the ranking of web search results in realistic settings.

Closely related to our work, Joachims [11] collected implicit measures in place of explicit measures, introducing a technique based entirely on clickthrough data to learn ranking functions. Fox et al. [9] explored the relationship between implicit and explicit measures in Web search, and developed Bayesian models to correlate implicit measures and explicit relevance judgments for both individual queries and search sessions. This work considered a wide range of user behaviors (e.g., dwell time, scroll time, reformulation patterns) in addition to the popular clickthrough behavior. However, the modeling effort was aimed at predicting explicit relevance judgments from implicit user actions and not specifically at classifying “best bets” via user behavior. Other studies of user behavior in web search include Lee et al. [14] and Rose et al. [19], but were not directly applied to improve ranking.

More recently, Joachims et al. [12] presented an empirical evaluation of interpreting clickthrough evidence. By performing eye tracking studies and correlating predictions of their strategies with explicit ratings, the authors showed that it is possible to accurately interpret clickthroughs in a controlled, laboratory setting. Unfortunately, the extent to which previous research applies to real-world web search is unclear. At the same time, while recent work (e.g., [21, 22, 24]) on using clickthrough information for improving web search ranking is promising, it captures only one aspect of the user interactions with web search engines, and has not investigated the effectiveness of these methods for ranking in the production settings. We build on existing research to develop robust user behavior interpretation techniques for the real web search setting. Instead of treating each user as a reliable “expert”, we aggregate information from multiple, unreliable, user search session traces. To the best of our knowledge, ours is the first work in the literature on effectively identifying “best bet” results in tandem with identifying the appropriate queries using past user behavior patterns.

8. CONCLUSIONS

The accuracy of the top result in web search is an important and challenging problem. We have empirically showed the effectiveness of exploiting the "wisdom of crowds" as features in our general learning framework, to select the most promising top result for a large and important class of web search queries.

Our large scale experiments on real user behavior data and queries demonstrate a significant improvement in accuracy due to using general user behavior models, with dramatic accuracy improvements when explicitly focusing on the "best bet" result identification problem. Specifically, we showed that the best bet problem in information retrieval can be more effectively solved using *classification* rather than the standard ranking approach.

Furthermore, we demonstrated the value of a general data mining and machine learning approach that achieves accuracy comparable to a domain-specific heavily engineered solution. Our general BehaviorClassifier method achieves significantly higher recall than the manually engineered solution (i.e., coverage of queries for which best bet results can be accurately proposed). As we discussed, the classification framework we developed is more amenable to easy maintenance and updating than an engineered approach, allowing our system to be easily tuned with evolving user behavior patterns and query distributions.

An important side effect of our approach is a general way of classifying user behavior as a first step towards better understanding user intent. So far, by construction, our focus was on navigational queries – i.e., queries for which a clear and obvious "best bet" results exist. However, many common queries are not navigational [5], and there may be multiple results for a user to examine to get all the necessary information. Our behavior classification method allows us to automatically detect these different types of behavior (e.g., Navigational vs. other) and hence can help identify the user intent even if there is no "best bet" result that can be returned. A promising direction is how to extend and apply these models to other types of queries, and how to improve user behavior modeling techniques to further improve the web search experience.

ACKNOWLEDGMENTS

We thank Eric Brill, Susan Dumais, Robert Ragno, Rohit Wad, and Ramez Naam for insightful comments and suggestions. We also thank Chris Burges and Matt Richardson for providing an implementation of RankNet for our experiments.

REFERENCES

[1] E. Agichtein, E. Brill, S. Dumais, and R. Ragno, Learning User Interaction Models for Predicting Web Search Result Preferences, in the Proceedings of SIGIR, 2006

[2] E. Agichtein, E. Brill, and S. T. Dumais, Improving Web Search Ranking by Incorporating User Behavior Information, in the Proceedings of SIGIR, 2006

[3] R. Baeza-Yates and B. Ribeiro-Neto, Modern Information Retrieval, Addison-Wesley, 1999.

[4] S. Brin and L. Page, The Anatomy of a Large-scale Hypertextual Web Search Engine, in the Proceedings of WWW, 1997

[5] A. Broder, A taxonomy of web search, SIGIR Forum, 2002

[6] C. J.C. Burges, T. Shaked, E. Renshaw, A. Lazier, M. Deeds, N. Hamilton, G. Hullender, Learning to Rank using Gradient Descent, in the Proceedings of ICML, 2005

[7] D.M. Chickering, The WinMine Toolkit, Microsoft Technical Report MSR-TR-2002-103, 2002

[8] M. Claypool, D. Brown, P. Lee and M. Waseda. Inferring user interest. IEEE Internet Computing, 2001

[9] S. Fox, K. Karnawat, M. Mydland, S. T. Dumais and T. White. Evaluating implicit measures to improve the search experience. ACM TOIS, 2005

[10] J. Goecks and J. Shavlick. Learning users' interests by unobtrusively observing their normal behavior. In the Proceedings of the IJCAI Workshop on Machine Learning for Information Filtering. 1999.

[11] T. Joachims, Optimizing Search Engines Using Clickthrough Data, in the Proceedings of SIGKDD, 2002

[12] T. Joachims, L. Granka, B. Pang, H. Hembrooke, and G. Gay, Accurately Interpreting Clickthrough Data as Implicit Feedback, in the Proceedings of SIGIR, 2005

[13] Kelly, D. and Teevan, J., Implicit feedback for inferring user preference: A bibliography. In SIGIR Forum, *2003*

[14] U. Lee, Z. Liu, J. Cho, Automatic Identification of User Goals in Web Search, In Proceedings WWW, 2005

[15] M. Morita, and Y. Shinoda, Information filtering based on user behavior analysis and best match text retrieval. In Proceedings of SIGIR, 1994

[16] D. Oard and J. Kim. Implicit feedback for recommender systems. In the Proceedings of the AAAI Workshop on Recommender Systems. 1998

[17] D. Oard and J. Kim. Modeling information content using observable behavior. In Proceedings of the 64th Annual Meeting of the American Society for Information Science and Technology. 2001

[18] F. Radlinski and T. Joachims, Query Chains: Learning to Rank from Implicit Feedback, Proceedings of the ACM Conference on Knowledge Discovery and Data Mining (KDD), ACM, 2005.

[19] D. E. Rose and D. Levinson, Understanding user goals in web search, In Proceedings of WWW 2004

[20] G. Salton & M. McGill. Introduction to modern information retrieval. McGraw-Hill, 1983

[21] X. Shen, C. Zhai, Active Feedback in Ad Hoc Information Retrieval, in Proceedings of SIGIR, 2005

[22] X. Shen, B. Tan, C. Zhai, Context-Sensitive Information Retrieval with Implicit Feedback, in Proceedings of SIGIR, 2005

[23] E.M. Voorhees, D. Harman, Overview of TREC, 2001

[24] G.R. Xue, H.J. Zeng, Z. Chen, Y. Yu, W.Y. Ma, W.S. Xi, and W.G. Fan, Optimizing web search using web click-through data, in Proceedings of CIKM, 2004