Mining Information Needs from Search Logs and its Application to Web Document Retrieval

ABSTRACT

This paper is concerned with automatic identification of queries with similar information need from search logs. Previous work often regards a query as a unique information need but users do issue all kinds of queries to look for the same target due to the intrinsic ambiguity and flexibility of natural languages. Some work clusters queries based on co-clicks; however, the information needs of queries in one cluster are not that same but roughly related. It is desirable to conduct automatic mining of queries with almost same information needs from a large scale search logs. In this paper, we take account of similarity between query strings. There are two issues associated with such similarity: it is too costly to compare any pair of queries in large scale search logs, and two syntactic similar queries, such as "SVN" and "SVM", are not necessarily similar in information needs. To address these issues, we propose to use similarity of query strings above the co-click based clustering results. Our method significantly improves precision over the co-click based clustering method (about 28% improvements in precision), and significantly outperforms a commercial search engine's query alteration (about 10% in F1 score). As application, we consider web document retrieval. We aggregate similar queries' click-throughs with the query's and evaluate them on a large scale dataset. Experimental results indicate that our proposed method significantly outperforms the baseline method of using a query's own click-throughs in all metrics.

1. INTRODUCTION

In this paper we address the issue of automatically identifying queries with the same information need from large scale search logs.

In traditional IR, a query is often regarded as a unique information need. However, we observe that users may use hundreds of different query strings to search the same target. For example, 561 unique queries about "environmental protection agency" appear in three-month Bing search logs. There are also more than 102 unique queries are about "pluto". As shown in Table 1, it is far from precise if we regard them as different requirements.

Only a few work pays attention to this issue. For example, query alterations (cite) are developed to help users input correct words and search relevant results. The technology is widely applied by search engines online, but it emphasizes much more on precision than recall. Clustering queries by co-clicks proves effective to improve context-aware query suggestion [5]. The queries sharing co-clicks are somehow related but do not necessarily share the same information need. For example, the cluster of "epa" also contains "epa address", "environmental", "director of epa", etc.

In fact, the queries with similar information needs look similar to each other. Some have mispellings; some have abbrievations; some contain a few stopwords; some contain the same keywords but in different order. One question arises here: can we identify queries with almost same information need and use them in web applications? We address the problem in this paper.

To the best of our knowledge, no previous work has been conducted on exactly this problem. It is not easy to cluster queries in terms of query strings' similarity from large scale search logs as it appears to be. First, it is not scalable to compare any pair of queries when the number of queries is huge in search logs. Calculating string similarity is expensive. Second, if no context, we have no confidence to infer which queries having the same information need. For example, "droken" is "draken" rather than "broken".

We take a two-step approach to address the problem. First, we apply the semantic clustering method proposed in [5] to get related queries based on co-clicks. Then, within each co-click based cluster, we propose a syntactic clustering method based on query strings to identify information needs. We evaluate the methods on a dataset, in which queries with the same information needs are manually labeled. The proposed semantic plus syntactic approach achieves as high precision as 93%. When compared to a commercial search engine's query alteration results, the approach improves recall by 36 points at the expense of only 2 points in precision.

We also propose two strategies to aggregate similar queries' click-throughs with the query's, i.e., one replacing strategy and one smoothing strategy. we apply a Borda Count fusion [15] to combine click-throughs with the widely used BM25 model [19]. Experimental results indicate that leveraging click-throughs of similar queries can effectively alleviate the sparsity of click-through data. The proposed approach significantly improves the baseline method in terms of NDCG@1, @3, and @10. The co-click based method is

SIGIR'14, July 6-10, 2014, Gold Coast, Australia.

Table 1: Examples of different queries with similar information needs

Query (Frequency in 12-week search logs)
epa (26529), environmental protection agency (4649), ..., united states environmental protection agency (319), ..., us environmental protection agency (238), ..., enviromental protection agency (183), ..., united states epa (35), ..., environmentalprotectionagency (17), environmental protective agency (10), ..., epa us (6), ..., united states environmental prtection agency (1), ..., enviromental protection in the united states 1
pluto (11078), planet pluto (1000), ploto (261), what is pluto (137), ..., plouto (50), ..., planit ploto (2), pluto and planets (2), the planets pluto (2), piut o (2), ..., teh planet pluto (1) ...

less effective because it brings more noise.

In the rest of the paper, we review related works in Section 2. In Section 3, we describe our approach to mine similar queries. Section 4 gives details on how we integrate click-throughs of similar queries into ranking models. Section 5 gives our experimental results. We make concluding remarks in Section 6.

2. RELATED WORK

Our work is mainly related to the following areas: query reformulation/alteration, query clustering, and click-throughs.

Query reformulation/alteration technologies usually change query terms for reducing the mismatch between the queries and documents to achieve the maximum relevance[8]. Most works like [7, 17, 2, 9, 4] focus on spelling correction. Gao et al [9] build the n-gram language model in the source channel model [18]. Wang et al [24] look into a query and mine the term level relevance for improving query reformulation. Guo et al [13] incorporate the character level, term level and query level mismatches into a unified CRF model for reformulating queries. Dang et al [8] propose to enhance reformulation relevance by applying manual anchors. This approach can reduce the query level ambiguity while has the same problems of scalability.

Some works investigate query clustering different applications [3, 25]. Wen et al [25] utilize the cross reference between queries and click documents to cluster queries, which expands measuring query similarity based on query strings. Cao et al [5] propose a co-click based clustering algorithm that is scalable to Web search logs. They cluster queries into concepts to solve the sparsity issue of session logs. Then they successfully apply the concepts in contextual query suggestion based on session data. There are also hybrid approaches [20, 22]. Sadikov et al[20] propose a Markov model to combine document and session features in predicting user intents. We adopt the clustering algorithm proposed in [5] as the first step of our proposed approach. Above the co-click based clusters, we apply similarity of query strings to identify queries with almost same information needs.

A lot of previous works are related to using click-through data for ranking Web documents. We can divide them into two groups. One group of works take advantage of click-throughs as training data to learn ranking models. Joachims et al. [16] introduces a novel retrieval model called Ranking SVM and trains models based on the relevance pairs of documents according to clicks. Some works, such as [6, 29, 28, 12] propose several click models to infer more trustworthy relationship between URLs from search logs. For example, Zhang et al. [28] incorporate queries and click-throughs in a session as a whole into training an extensive click model for understanding extended user behaviors in a collaborative way. The other group of works integrate click-through information into ranking functions or models. Xue et al. [27] extract queries for each click in search logs and build them as a metadata of Web document. Their experimental results indicate that adding such a metadata significantly improves retrieval performance measured by Precision@20. Agichtein et al. [1] incorporate user click-through data into implicit user feedback model for ranking, which showed a significant improvement of retrieval than the traditional state-of-the-art content based approaches. Our work is in the second group, but our focus is to enrich click-throughs by mining similar queries, rather than investigate how to integrate relevance features from click-throughs.

In addition, sparsity of click-throughs is a well-known problem [11]. As most queries have limited clicks or even do not have clicks, the works on using click-throughs cannot essentially influence many queries. Some ideas have been proposed to address the sparsity problem. Wu et al. [26] apply a kernel method that takes the query similarity valued by co-clicked documents to deal with mismatch between a given query and queries associated with a document. The experimental results show a strong improvement in terms of NDCG@1, NDCG@3 and NDCG@5 over BM25. Song et al. [21] introduce the skip information and build a skip graph for enriching user behavior features. The work also compensates the lacking-click problem on rare queries. Gao et al. [10] propose a smoothing approach to expand query-document features by predicting missing values of documents using click streams. These kinds of works measure query similarity based on co-clicks. In this paper, we experiment with both a method based on co-clicks and another method that takes account of similarity of query strings in additional to co-clicks. In experiments, we find that some similar queries based on co-clicks may drift from the topic of a query and thus bring some noisy click-throughs.

3. MINING INFORMATION NEEDS

In this section, we describe our proposed two-step approach to mine queries with same information needs. The first step measures query similarity based on co-clicks. We adopt the implementation of query clustering proposed by previous work [5]. The second step further considers query strings in measuring similarity between queries. We propose applying the second step within clusters we obtain via the first step. Thus, we can afford the complexity of the second step and ensure quality.

3.1 Clustering Queries based on Co-Clicks

In this paper, we process the queries with at least one click, while ignoring the queries without clicks because there is no enough evidence to infer their information needs.

Similar to [5], we build a click-through bipartite graph $G = (V_q, V_u, E)$ from search logs, where V_q is the set of query nodes, V_u is the set of URL nodes, and E is the set

Table 2: Cases where the CC+BM25 method performs worse than the FM+BM25 method

CC method		FM method	
Rank	URL (Rating)	Rank	URL (Rating)
Query: alabama adventureland (3)			
Similar	Similar: alabama adventure (1052), adventure park in alabama (2), alabama adventures (138), water parks in alabama (98), amusement park alabama (4), visionland birmingham alabama (23), alabama adventure employment (2), ...		alabama adventure land (12), alabama adventurland (1)
1	http://alabamaadventure.com (Fair)	1	http://alabamaadventure.com (Fair)
2	http://themeparkcity.com/usa_al.htm (Fair)	2	http://adventurelandthemepark.com (Perfect)
Query: blue planet (987)			
Similar	www blueplanetbiomes org (698), blueplanetbiomes org (582), blueplanet (202), blue planet com (96), discovery blue planet (3), which planet is blue (1), the blue planet show times (3), the blue planet video (1), ...		blueplanet (202), blueplant (31), blueplannet (3), bleplanet (1), blue planeht (1), blue planert (1), blue planlet (1)
1	http://blueplanetbiomes.org (Fair)	1	http://dsc.discovery.com/.../blue-planet.html (Perfect)
2	http://dsc.discovery.com/.../blue-planet.html (Perfect)	2	http://blueplanetbiomes.org (Fair)

of edges. An edge e_{ij} is connected between a query node q_i and a URL node u_j if u_j has been clicked when users issued q_i. The weight w_{ij} of edge e_{ij} is the aggregated click times. Then the query q_i is represented as an L_2-normalized vector, where each dimension is one URL. If edge e_{ij} exists, the value of the dimension is $norm(w_{ij})$; otherwise, it is zero. We also apply Euclidean distance to calculate the distance between two queries.

We apply the clustering method proposed in [5] in mining query concepts or clusters in this paper. The algorithm creates a set of clusters when it scans the queries. For each query q, the algorithm first finds the closest cluster C, and then test the diameter of $C \cup \{q\}$. If the diameter is not larger than D_{max}, q is assigned to C, which is updated then. Otherwise, a new cluster is created for q.

The distance between a query q and a cluster C is given by

$$distance(q, C) = \sqrt{\sum_{u_k \in U} (\vec{q}[k] - \vec{c}[k])^2}$$

where, $\vec{c} = norm(\frac{\sum_{q_i \in C} \vec{q_i}}{|C|})$ is the normalized centroid of the cluster and $|C|$ is the number of queries in C.

And the diameter measure is defined as

$$D = \sqrt{\frac{\sum_{i=1}^{|C|} \sum_{j=1}^{|C|} (\vec{q_i} - \vec{q_j})^2}{|C|(|C| - 1)}}$$

The diameter is ranged from 0 to $\sqrt{2}$. We set D_{max} as 1 in our experiments as suggested by [5]. More details of the clustering algorithm can be found in the paper.

Although the algorithm is efficient and scalable, we cannot afford clustering all queries in the three-month logs due to time and space cost. Please note we do not prune any queries or clicked URLs in order to keep related queries for rare queries. Therefore, we parallelize the process of clustering queries.

Table 2 and Table 3 show four example clusters at the left side. A cluster is usually related to one concept. For example, the first cluster Table 3 contains various queries on PayPal and the second cluster contains queries on Michael Jackson's death. In such cases, click-throughs of similar queries may help increase relevant clicks for a query. Sometimes a cluster may contain several different subtopics about one central concept. For example, the first cluster in Table 2 contains queries on parks in Alabama, such as adventure theme park, water park and visionland park, and related information about the parks, such as employment or hours. In such a case, click-throughs of other queries in a cluster may be irrelevant to a query.

3.2 Grouping queries based on Fuzzy Match

To address the issue that queries in a co-clicks cluster may be less relevant to each other, we propose the second step, i.e. grouping queries within each cluster based on fuzzy match of query strings. Query similarity based on keywords is not new, but no previous work applies it to large scale query clustering due to complexity. We solve the efficiency problem by applying the fuzzy match based method within clusters that we obtain by applying the co-clicks based method.

We observe that misspelling often occurs in real users' query logs, e.g., "planet", "planeht" and "planert". Previous research reveals that there are approximately 10-15% of the queries issued to search engines contain misspellings [7]. Furthermore, users can use different words with the same stem to describe the same thing in English, e.g., "adventure" and "adventures". Some new compound words appear on the Web and users are not sure about their spellings, e.g., "paypal" or "pay pal". As queries in a cluster have already had some common clicks, we are more confident that some of them are seeking same targets if their appearances are similar enough.

Based on our observations, we propose using character level similarity based on weighted Levenshtein distance [23] in our fuzzy match step.

First, we build a graph based on query nodes in a cluster:

> For each node $n, n' \in C$, an edge is added edge between n and n' if $dist(n, n') < \theta_{hard}$ or $dist_s(n, n') < \theta_{soft}$

where $dist(n, n')$ means the weighted Levenshtein distance between query strings of n and n', and $dist_s(n, n')$ is defined

Table 3: Cases where the (Q,CC)+BM25 method performs better than the (Q,FM)+BM25 method

(Q,CC)+BM25 method		(Q,FM)+BM25 method	
Rank	URL (Rating)	Rank	URL (Rating)
Query: paypal (11367)			
Similar	paypal com (16766), www paypal com (4145), pay pal (1603), pay pal com (394), paypal plus (326), palpal com (265), ...		pay pal (1603), paypall (233), pypal (48), paypale (31), patpal (29), pau pal (26), pay pl (24), ...
1	https://paypal.com (Perfect)	1	http://en.wikipedia.org/wiki/paypal (Excellent)
2	http://en.wikipedia.org/wiki/paypal (Excellent)	2	http://paypal.com.au/au (Fair)
Query: michael jackson is dead (33)			
Similar	michael jackson dies (100), how michael jackson died (94), when michael jackson died (62), where michael jackson died (12), mickal jackion picture (1), ...		micheal jacksons dead (3), michael jacksons dead (2), michaels jacksons death (2), miichael jacksons death (2), michaels jackson s death (1), ...
6	http://en.wikipedia...michael_jackson (Excellent)	8	http://abcnews.go.com/...&page=1 (Good)
7	http://abcnews.go.com/...&page=1 (Good)	12	http://en.wikipedia...michael_jackson (Excellent)

as follows:

$$dist_s(n, n') = \frac{2dist(n, n')}{length(n) + length(n')}$$

We set two kinds of distance thresholds in our algorithm. The first θ_{hard} is called the *hard distance threshold*, which limits the maximum character difference between two queries. This threshold mainly helps coalesce short queries. The second θ_{soft} is called the *soft distance threshold*, which related to the length of the two queries. This threshold mainly helps coalesce long queries because it can tolerate more character difference.

Then, the graph is partitioned into connecting sub-graphs, and the nodes in each connecting sub-graph are regarded as similar queries.

Table 2 and Table 3 show four examples at the right side. Compared to the co-clicks based results, the similar queries identified by this step more likely stick to a narrow information need. For example, there are only two similar queries to "alabama adventureland". There is slight difference in spelling between them. In the example of "blue planet", the queries on "blueplanetbiomes" are also excluded from similar queries. Such changes tend to raise the relevance of click-throughs from similar queries.

As building the query graph requires one to one query comparison, time complexity is $O(N^2)$, where N is the number of queries in cluster C. As the size of clusters is in power-law like distribution (we will show the distribution in Section 5.2.1), most of clusters have a small number of queries. Furthermore, the maximum size of cluster is much less than the number of all queries in logs and can be controlled in the co-clicks based method. Therefore, the time complexity of the fuzzy match based method is acceptable for very large scale query logs.

4. WEB RETREIVAL METHOD

Previous work has demonstrated that click-throughs are effective to improve Web search [16, 27, 1]. Instead of complex models, we apply a simple but conventional way, i.e., Borda Count, to fuse click-throughs with a baseline ranking function based on content, i.e., the BM25 model. We aim to show the pure impact of enhancing click-throughs data by the mined information needs.On aggregating click-throughs, we describe two strategies: a) replacing a query's click-throughs by similar queries' and b) smoothing a query's click-through by similar queries'.

4.1 Integrating Click-Throughs in Ranking

Click-through data is regarded as implicit feedback from users. In general, the number of clicks for URLs is positively associated with relevant degrees of documents or how desired documents are by Web users. For example, the URL that has been clicked most for a query is usually the most relevant to the query or what users most want. Thus, almost all commercial search engines make use of clicks to improve their ranking functions.

In this paper, we take the widely used ranking model BM25 as our baseline ranking function based on content of Web documents. We retrieve top 20 documents using the BM25 formula based on multiple fields [19] from Bing Search's index. Each Web document contains four fields: anchor, title, body and URL. These documents are regarded the most relevant ones based on content.

When we have a list of URLs sorted by clicks, we first filter those URLs that are not among the top 20 documents returned by our baseline ranking function. Then we apply Borda's fusion method [15] to combine the two lists:

$$Score(c) = \lambda \frac{1}{r_b(d)} + (1 - \lambda) \frac{1}{r_c(d)} \quad (1)$$

where r_b is the rank of document d in the list of top 20 documents returned by the BM25 formula, and r_c is the rank of d in the list of clicked documents that are also among the former list. λ is a trade-off parameter to balance two lists. In our experiments, we find that 0.9 is the optimal λ for all methods and we set λ as 0.9.

4.2 Aggregating Click-Throughs

Given a query q, we denote a set of similar queries that are mined by an algorithm a as $S_a(q)$. Then, we can aggregate all clicks for queries in the set to a list of clicked URLs:

$$c(S_a(q), u_i) = \sum_{c(q_j, u_i)!=0 \& q_j \in S_a(q)} c(q_j, u_i);$$

One strategy is using click-throughs of similar queries, i.e., $c(S_a(q), u_i)$ to replace click-through of the given query, i.e., $c(q, u_i)$. We call it *Replacing* strategy. The other strategy is *Smoothing* stategy. We smooth a query's click-throughs as follows:

$$c_s(q, S_a(q), u_i) = \beta \frac{c(q, u_i)}{\max_{u_j} c(q, u_j)} + (1-\beta) \frac{c(S_a(q), u_i)}{\max_{u_k} c(S_a(q), u_k)}$$

In our experiments, we have tried the above two strategies of using click-throughs of similar queries.

5. EXPERIMENTS

5.1 Setup

We extract Bing Search logs of 12 weeks from January 1 to March 25, 2010. All queries with at least one click are used. There are about 506 million unique queries in the 12-week logs. We also have the information of every click, i.e., <query, query time, clicked URL>. Thus, by aggregating, we can obtain the tuple of <query, clicked URL, #clicks> for each query. We apply the algorithms that described in Section 3 to these data and mine similar queries. To our best knowledge, the scale of logs is the largest among related works.

5.1.1 Evaluation of Query Clustering

To evaluate the proposed approach, we create a dataset of 100 sampled clusters, in each of which the queries with same information needs are manually labeled by assessors.

First, we randamly select 100 "trunk" clusters from the Bing Search logs in January 2009. We do not select "head" clusters which contain too many queries because it is difficult for assessors to handle. We do not select "tail" clusters which contain too few queries either because they provide less valuable cases for our investigated problem. The selected clusters contain 1,560 queries. The average number of queries in a cluster is about 16.

Then, we design an easy-to-use tool to assist assessors merge the queries with same information needs. Queries in a cluster are shown in a list at the left and each of them has a checkbox beside. An assessor can easily check the queries with same information needs and then all checked queries are shown as a merged group at the right. The tool also provides easy ways to modify existing query groups. Assessors work on the clusters after reading the guidelines of this task and exercising on five example clusters. They can also refer to search results anytime if they are not sure about the meanings of a query.

Finally, we obtain a labeled dataset, in which 589 query groups are manually created. The average number of information needs per cluster is about 6.

Given a set of queries, denoted by A, mined by an algorithm, we first align the set with the query groups manually labeled. The query group that has the maximum number of common queries with A is chosen as corresponding ground-truth information need set I. Then, precision and recall of A are defined as:

$$Precision(A, I) = \frac{|A \cap I|}{|A|} \quad (2)$$

$$Recall(A, I) = \frac{|A \cap I|}{|I|} \quad (3)$$

F1 measure is the geometric average of precision and recall.

Above the labeled dataset, an algorithm generate several A_i and we can align them with corresponding I_is. To average the metrics over the algorithm result, we can apply Micro or Macro average precision and recall as follows:

$$Micro\text{-}Precision(\{A_i\}_m, \{I_j\}_n) = \frac{\sum_{i=1}^{m} |A_i \cap I_i|}{\sum_{i=1}^{m} |A_i|} \quad (4)$$

$$Micro\text{-}Recall(\{A_i\}_m, \{I_j\}_n) = \frac{\sum_{i=1}^{m} |A_i \cap I_i|}{\sum_{i=1}^{m} |I_i|} \quad (5)$$

$$Macro\text{-}Precision(\{A_i\}_m, \{I_j\}_n) = \frac{1}{m} \sum_{i=1}^{m} \frac{|A_i \cap I_i|}{|A_i|} \quad (6)$$

$$Macro\text{-}Recall(\{A_i\}_m, \{I_j\}_n) = \frac{1}{m} \sum_{i=1}^{m} \frac{|A_i \cap I_i|}{|I_i|} \quad (7)$$

Intuitively, the Micro metrics regard each query as a basic unit and assign equal weights in averaging, whereas the Macro metrics regard each query set A_i as a basic unit and assign qual weights.

5.1.2 Evaluation of Document Retrieval

We use a large scale query set that contains 13,920 queries for evaluating retrieval effectiveness. On average, there are more than 100 documents are judged in five-grade relevance levels, i.e., perfect, excellent, good, fair and bad, corresponding to the ratings from 5 to 1. When we intersect the query set with our processed 12-week query logs, there are 7,453 common queries. We conduct ranking experiments on these 7,453 queries and their corresponding relevance assessments of documents.

We apply widely used NDCGs (Normalized Discounted Cumulated Gain) [14] as evaluation metrics of retrieval. The five relevance ratings have gains of $2^{rating} - 1$. We report NDCGs at three cutoffs, i.e. 1, 3, and 10. Among them, NDCG@1 and NDCG@3 mainly measure top performance, which is highly related to users' satisfaction at the first glance, whereas NDCG@10 measures deeper performance, which corresponds to the first page of search results.

5.2 Statistics

5.2.1 On Query Clusters

We plot query frequencies in a descending order in Figure 1. The axis could stand for queries, fuzzy matched based information need groups, or co-click based clusters. The query frequency of a group is defined as the sum of frequencies of all queries belonging to the group. We call the query frequency of a group as group frequency for short. Cluster frequency is defined in a similar way. To be simple, query frequency refers to the frequency based on queries in the remaining parts.

Previous works discovered that frequencies based on queries are in a Zipf distribution. As shown in Figure 1(a), a long tail corresponds to the huge amount of queries that have been issued only once in the 12 weeks. When we group queries with similar information needs together, the distribution of group frequency is also Zipf (See Figure 1(b)); however, more queries move close to y axis with their groups. As a result, the long tail is not so long as that of query frequency. In our experiments, the number of groups is about 83% of the number of unique queries. This indicates that the information needs behind some long tail queries are not that rare. They may be variants of head or body queries. Similar to group frequency, cluster frequency distribution has an even shorter tail as shown in Figure 1(c). The number of clusters is about 46% of the number of queries. On average, one co-click based cluster is split into 1.8 query groups.

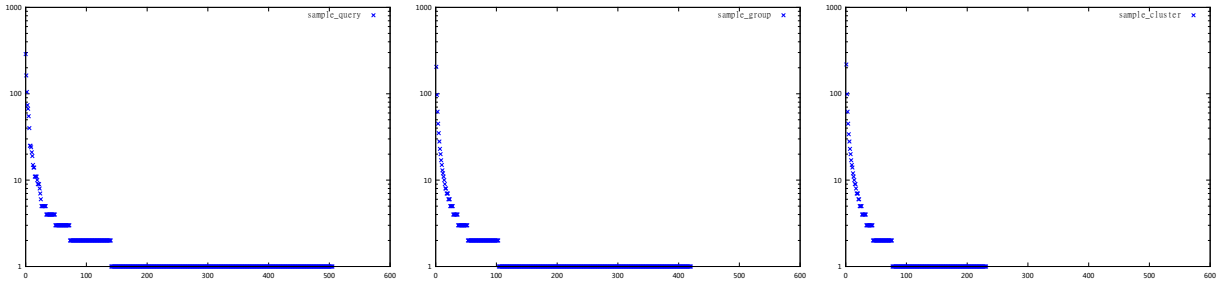
5.2.2 On Click-Throughs

Table 4: Compare the amount of similar queries and click-throughs per query on our query set

	#similar	#urls	times over Q	#clicks	times over Q
Q	1	39	1	21,105	1
FM	89	75	1.94	70,455	3.34
CC	427	256	6.60	97,801	4.63

Table 5: Comparing retrieval effectiveness of the methods that are using click-throughs of similar queries based on different mining algorithms. Notes: We apply t-test in significant test. The baseline is Q+BM25. † means the difference is statistically significant with p-value less than 0.01; * means the difference is statistically significant with p-value less than 0.05.

Methods	NDCG@1	Imp.	NDCG@3	Imp.	NDCG@10	Imp.
BM25	54.16	-15.86%†	51.05	-9.55%†	51.22	-5.77%†
Q+BM25	64.37	0.00%	56.45	0.00%	54.35	0.00%
CC+BM25	63.69	-1.06%†	56.12	-0.58%†	54.30	-0.09%
FM+BM25	64.49	0.18%	56.53	0.16%*	54.43	0.15%†
(Q,CC)+BM25	64.36	-0.01%	56.47	0.04%	**54.45**	**0.19%**†
(Q,FM)+BM25	**64.57**	**0.31%**†	**56.54**	**0.16%**†	54.43	0.15%†



(a) Queries (b) Groups (c) Clusters

Figure 1: Frequency distribution based on different granularity

We do statistics of similar queries and click-throughs over the 7,543 queries that are used for evaluating web retrieval effectiveness. We average the number of similar queries, the number of clicked URLs and the number of clicks. Results are shown in Table 4. On average, a query has 39 clicked URLs and 21,105 clicks. The numbers are huge because we accumulate about three-month logs from a popular commercial search engine.

According to Table 4, our proposed approach using fuzzy match (FM) mines about 89 similar queries per query and the method based on co-clicks (CC) finds even more, i.e., 427 similar queries. Thus, aggregating click-throughs of similar queries dramatically increase the number of clicked URLs and the number of clicks over click-throughs of the query. The CC method increases the number of clicked URLs increases to 6.6 times and the number of clicks to 4.6 times. As expected, the FM method increases fewer click-throughs than the CC method, but the number of clicked URLs is still increased by about one time and the number of clicks is increased by more than two times.

5.3 Query Clustering Evaluation Experiment

We evaluate the groups that our proposed approach generate (denoted as Group) and compare the groups with the method regarding each query as an individual information need (denoted as Query), and the clusters based on co-clicks (denoted as Cluster). Results are shown in Table 6. It is not surprising that Query obtains the 100% precision as each query set A_i contains one query only, and Cluster obtains the 100% recall as we create the dataset by asking assessors dividing those clusters. However, the recall of Query is as low as 0.215 and the precision of Cluster is as low as 0.638 in Micro-Precision or 0.719 in Macro-Precision. Our proposed groups significantly increase the clusters in precisions. It achieves as high as 0.931 Micro-Precision and 0.978 in Macro-Precision. This indicates that our proposed method is effective to identify the queries with quite similar informatin needs in a cluster.

Table 6: Evaluating different methods over manually labeled data

Metrics	Query	Group	Cluster	Bing
Micro-Precision	1	0.931	0.638	0.952
Micro-Recall	0.215	0.613	1	0.451
Micro-F1	0.354	0.739	0.779	0.612
Macro-Precision	1	0.978	0.719	0.986
Macro-Recall	0.215	0.484	1	0.390
Macro-F1	0.354	0.647	0.836	0.559

Previous works on query alterations is highly related to our proposed approach. We conduct an experiment to compare our proposed method with a commercial search engine Bing Search's alterations. As we know, Bing Search inte-

grates some state-of-the-art query reformulation and alteration models and tunes the models on more data in terms of both the ammount of data and the number of resources.

We obtain all alterations of the queries in the labeled dataset from Bing Search. In additional to these, we also apply stopwords filtering and ignoring term order of keywords, and thus the results would be improved in our dataset. Such comparison would be more fair. The result of alteration is shown in the last column of Table 6. We find that query alterations of Bing Search perform slightly better than our proposed groups in precisions, i.e., two points gap in Micro-Precision and one point gap in Macro-Precision; whereas, our proposed approach performs significantly better than the query alterations in recalls, i.e., 16 points gap in Macro-Recall and 9 points in Macro-Recall. Overall, in terms of F1 measure, our proposed approach outperforms Bing by about 10 points.

We look closely into the generated data and find some reasons that result in the lower recall of alterations. As the query alteration technology is applied to online queries, it has to be cautious to ensure high precision because users will be dissappointed if the alterations are misleading. Usually the search engines leverage machine learning technologies to predict the confidence that an alteration is right for one individual query word or occasionally a query phrase and output only the alterations with enough confidence. The predictors are optimized globally. In contrary, we have co-click based clusters first and leverage the rich local information within a cluster to mine the knowledge on misspellings, acronyms, etc. We can trust some queries with low frequency and merge them into an appropriate query group too. Furthermore, our proposed method is applied offline and it can afford more expensive calculation. Therefore, our proposed method does much better job than query alterations in recalls.

5.4 Web Document Retrieval Experiment

We evaluate the ranking results retrieved by six methods: the BM25 method (BM25), the method combing query clicks with BM25 (Q+BM25), the method combining aggregated clicks based on co-clicks with BM25 (CC+BM25), the method combining aggregated clicks based on fuzzy match with BM25 (FM+BM25), the method combining smoothed clicks based on co-clicks with BM25 ((Q,CC)+BM25), and the method combining smoothed clicks based on fuzzy match ((Q,FM)+BM25). Results are shown in Table 5.

5.4.1 Analysis on Replacing Strategy

The third and fourth methods repace a query's clicks by the aggregated clicks mined from the CC method, which clusters queries based on co-clicks [5], and the FM method, which further calculates similarity between queries based on query strings. Although the CC method finds much more similar queries, combining the corresponding aggregated clicks with BM25 performs worse than the Q+BM25 method, our baseline method, in all NDCGs. The difference is significant in NDCG@1 and NDCG@3. This indicates that the CC method may bring some clicks that hurt retrieval performance. The FM method is more careful in mining similar queries. As the table shows, combining the aggregated clicks with BM25 outperforms the baseline Q+BM25 method in all NDCGs. The difference is significant in NDCG@3 and NDCG@10.

We conduct a case study and show two example queries in Table 2. The number in parentheses behind a query is the number of summed clicks for the query. When looking close to the similar queries, we find that the CC method may cluster related queries, but some of them may have different information needs from the given query. For one instance, besides similar queries on adventure park in alabama, the CC method also mines some other parks in alabama, such as water parks, amusement park and visionland, and some other subtopics about adventure park, such as employment and hours. By aggregating clicks for these queries, the url `http://themeparkcity.com/usa_al.htm` is boosted to No.2, although it is only fairly relevant to the query "alabama adventureland". On the contrary, the FM method only returns two queries that are very similar to the given query. And thus aggregating clicks from those queries boosts a perfect URL `http://adventurelandthemepark.com` to No.2 search result. In such a case, NDCG@3 increases when the FM method is applied while it decreases when the CC method is applied. The example query "blue planet" with more clicks is similar to this case.

5.4.2 Analysis on Smoothing Strategy

The fifth and sixth methods smooth query clicks by aggregated clicks of similar queries and then combine the smoothed clicks with BM25. As Table 5 shows, the two methods perform better than corresponding methods using only aggregated clicks of similar queries respectively. Compared to the baseline Q+BM25 method, the (Q,CC)+BM25 method is significantly better in terms of NDCG@10, although there is no significant improvement in terms of NDCG@1 and NDCG@3. It indicates that the CC method may bring less relevant clicked URLs because of not-very-similar queries. Such URLs hurt top retrieval effectiveness while help the retrieval effectiveness of deeper cutoffs, in particular when we use the click-throughs to smooth query clicks.

The (Q,CC)+BM25 method is even better than the (Q,FM) +BM25 method, although the difference is not significant. We show such two cases in Table 3. For instance, for the popular query "paypal", the CC method can group some frequent queries about paypal.com and boosts the official website to No.1, whereas the FM method cannot discover these kind of queries and thus ranks the entry about paypal on wikipedia at No.1. Another instance is the query "michael jackson is dead". Although it is only clicked for 33 times, its similar queries that are mined by the CC method is much more popular and cover related subtopics like when/how/where Michael Jackson died. These related queries do not drift from the query and contribute clicks to the authoritative URL about death of Michael Jackson from wikipedia. As a result, the (Q,CC)+BM25 method performs better than the (Q,FM)+BM25 method in NDCG@10.

The evaluation results in Table 5 indicates that the (Q,FM) +BM25 method significantly outperforms the baseline Q+ BM25 method in all NDCGs. It is also the best among the six methods in terms of NDCG@1 and NDCG@3. This indicates that the queries mined by the FM method are effective in improving retrieval performance. We conduct a case study on why click-throughs of similar queries can improve retrieval effectiveness. Some cases are shown in Table 7. The number of clicks for a query is shown in the parentheses following the query. As the table shows, for a query with only a few clicks, such as "rice gardens", the fuzzy match method

Table 7: Cases where the (Q,FM)+BM25 method performs better than the Q+BM25 method

Method	(Q,FM)+BM25	Q+BM25
Query	rice gardens (1)	
Similar	rice garden (99), rice gardenb (1)	
No.1	http://thericegarden.com (Perfect)	http://allrecipes.com/.../detail.aspx (Bad)
No.2	http://ricegarden.biz/locations.php (Fair)	http://gardenrice.com (Bad)
No.3	http://ricegarden.biz/menu.php (Fair)	http://gardensofricecreek.com (Bad)
Query	g earth (100)	
Similar	googleearth (133088), googleearth com (127522), google earth com (58832), gogle earth (7832), goole earth (6700), google eart (5964), ..., googleeath (830), ...	
No.1	http://earth.google.com (Perfect)	http://earth.google...download-earth.html (Good)
No.2	http://earth.google...download-earth.html (Good)	http://earth.google.com (Perfect)
No.3	http://gearthblog.com (Good)	http://gearthblog.com (Good)
Query	greatest movie quotes (302)	
Similar	great movie quotes (1530), great moive quotes (3), greates movie quotes (2), greastest movie quotes (2), graet movie quotes (1), gret movie quote (1), grreat movie quotes (1), grwat movie quotes (1), great movie quoets (1), greata movie quotes (1), grreat movie quote (1)	
No.1	http://filmsite.org/greatfilmquotes.html (Good)	http://en.wikipedia.org/...quotes (Fair)
No.2	http://en.wikipedia.org/...quotes (Fair)	http://filmsite.org/greatfilmquotes.html (Good)
No.3	http://franksreelreviews.com/...rantquote.htm (Good)	http://franksreelreviews.com/...rantquote.htm (Good)
Query	free quicktime download (4122)	
Similar	free quick time download (398), free quick time down load (24), freequicktimedownload (16), free quicktime dowload (14), free quicktime downlaod (14), free quicktime downlode (8), fee quicktime download (6), free quictime downloads (6), ...	
No.1	http://apple.com/quicktime/download (Perfect)	http://quicktime-download.info (Good)
No.2	http://quicktime-download.info (Good)	http://apple.com/quicktime/download (Perfect)
No.3	http://softwarepatch.com/.../quicktime.html (Good)	http://softwarepatch.com/.../quicktime.html (Good)

can connect it to other queries with more clicks, such as "rice garden". A highly relevant document is then discovered based on richer clicks. For a query with middle number of clicks, such as "g earth" and "greatest movie quotes", the fuzzy match method can find more popular queries with similar search intents, such as "google earth" and "great movie quotes". When leveraging click-throughs from these similar queries, more authoritative URLs pop up at the top. The fuzzy match method is even helpful for the most popular queries, such as "free quicktime download". One reason may be that spams often follow the most popular queries and aggregating the click-throughs for different similar queries can somehow recover some really wanted URLs.

It is necessary to use the click-throughs of similar queries to smooth query clicks, rather than replacing query clicks. In other words, the click-throughs of similar queries are complementary to the click-throughs of a query. For example, as shown in Table 8, given a query "transformers 2", the FM method can mine some similar queries like "transformer". In fact, the movie Transformer 2 is the sequal of Transformer. Thus two homepages from IMDB website are discovered in click-throughs. When we aggregate all clicks of similar queries, the homepage for the movie Transformer wins. As a result, the Q+BM25 method ranks the good URL at No.2 while the FM+BM25 method ranks it at No.6, which is behind the bad URL. Fortunately, the method of smoothing query clicks can solve a part of such issues.

5.4.3 Analysis on Query Segments

To answer which queries benefit from click-throughs of similar queries, we further analyze the results of the best method (Q,FM)+BM25 and the baseline method Q+BM25. First, we sort all queries by the number of summed clicks in an ascending order and number them. Then, to ensure stable NDCGs, we average NDCGs of the first 2000 queries and calculate the relative improvement of the (Q,FM)+BM25 method over the baseline method on the 2,000 queries. We get the first three columns in Figure 2. The number of clicks for 2,000 queries ranges from 1 to 19. As the columns show, the improvement in NDCG@1 and NDCG@3 is about 0.5% and the improvement in NDCG@10 is about 0.33%. These three columns compose a group for the first query segment. Next, we do similar things to No.501 to No.2500 queries. There are still 2,000 queries in the second query segment and there are 1,500 overlapped queries with the first query segment. We get the second group of three columns, which stand for the relative improvement in NDCGs. Similarly, we calculate relative improvements for the other ten query segments. Each segment contains 2,000 queries. Two adjacent segments have 1,500 overlapped queries, except for the last two query segments. As we have 7,453 queries, the last query segment contains No.5454 to No.7453 queries and the second last segment contains No.5001 to No.7000 queries. The last two segments have 1,546 overlapped queries. Finally we get twelve group of columns for the query segments as shown in Figure 2.

In terms of NDCG@3, the improvement over query segments goes down with the number of clicks for a query increasing. This indicates that our mined similar queries and their click-throughs are most helpful for the queries with fewer clicks. It is reasonable because the queries with rich enough clicks have less space to be improved, in particular at the top. We conduct t-test over each segment and find that the improvement in NDCD@3 for the first three query segments is significant. Thus, we can conclude that our proposed method can significantly benefit the queries with the number of clicks from 1 to 72 (about 25% queries in our set). For the queries with more than 135 clicks, our proposed method cannot significantly improve retrieval effectiveness of top three documents.

Table 8: One case where the FM+BM25 method performs worse than the Q+BM25 method

Query: transformers 2 (6882)	Similar: transformer (5814), transformers2 (1995), transfomers (1383), ...
Q+BM25 method	FM+BM25 method
No.2 - http://imdb.com/title/tt1055369 (Good)	No.4 - http://imdb.com/title/tt0418279 (Bad)
No.6 - http://imdb.com/title/tt0418279 (Bad)	No.6 - http://imdb.com/title/tt1055369 (Good)

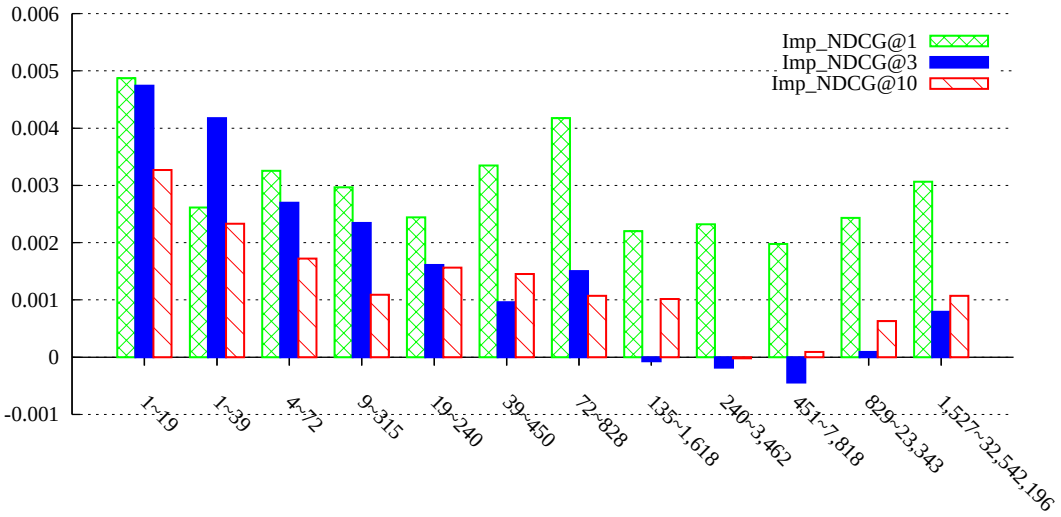


Figure 2: Analyze improvement of the (Q,FM)+BM25 method over the Q+BM25 method on different query segments divided by the total number of clicks for a query

In terms of NDCG@10, the improvement over query segments has a similar trend. A difference is that our proposed method obtains significant improvement in NDCG@10 over more query segments. They are No.1,2,3,4,6 and No.7 query segments. This indicates that the aggregated click-throughs can benefit more queries, with the number of clicks up to 828 (about 50% queries in our set), in retrieval effectiveness of the first search page.

In the figure, we also observe that there is improvement over all query segments in terms of NDCG@1. As NDCG@1 replies on the first returned document only, the change between two ranking results is usually large in value. We cannot draw a conclusion that our proposed method benefits all queries in top one retrieval performance. Such results still indicate that our proposed method is positive to enhance users' satisfaction.

6. CONCLUSION AND FUTURE WORK

In this paper, we have investigated the problem of automatically mining queries with almost same information need from large scale search logs, and have investigated how the mined similar queries can help improve web document retrieval.

We proposed a two-step appproach to address the problem. We implemented a scalable query clustersing method based on co-cliks. Then within each cluster, we group queries in terms fo syntactic similarity based on query strings. Experimental results indicate that our proposed approach significantly improves the precision of co-click based cluster by 30 points. It also outperforms Bing Search's query alteration by about 10 points in terms of F1 measure.

In the application of web retrieval, we propose to aggregate click-throughs of similar queries to address the sparsity issue. Experimental results indicate that smoothing a query's click-through by the aggregated data works better than the replacing strategy. When we taking the smoothing strategy, the mining method based on co-clicks cannot beat the baseline method in terms of NDCG@1 and NDCG@3, but it works the best in terms of NDCG@10. Based on the clusters, our proposed method using fuzzy match wins the baseline method in terms of all three NDCGs, and the differences are statistically significant. Through analysis over different query segments, we find that the similar queries' click-throughs benefit the queries with fewer clicks more than the queries with rich clicks.

7. REFERENCES

[1] E. Agichtein, E. Brill, and S. Dumais. Improving web search ranking by incorporating user behavior information. In *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 19–26. ACM, 2006.

[2] F. Ahmad and G. Kondrak. Learning a spelling error model from search query logs. In *Proceedings of the 5th conference on HLT and EMNLP*, pages 955–962. ACL, 2005.

[3] D. Beeferman and A. Berger. Agglomerative clustering of a search engine query log. In *Proceedings of the 6th international ACM SIGKDD conference on Knowledge discovery and data mining*, pages 407–416. ACM, 2000.

[4] E. Brill and R. Moore. An improved error model for noisy channel spelling correction. In *Proceedings of the*

38th annual meeting on ACL, pages 286–293. ACL, 2000.

[5] H. Cao, D. Jiang, J. Pei, Q. He, Z. Liao, E. Chen, and H. Li. Context-aware query suggestion by mining click-through and session data. In *Proceeding of the 14th international ACM SIGKDD conference on Knowledge discovery and data mining*, pages 875–883. ACM, 2008.

[6] O. Chapelle and Y. Zhang. A dynamic bayesian network click model for web search ranking. In *Proceedings of the 18th international conference on World wide web*, pages 1–10. ACM, 2009.

[7] S. Cucerzan and E. Brill. Spelling correction as an iterative process that exploits the collective knowledge of web users. In *Proceedings of EMNLP*, volume 4, pages 293–300, 2004.

[8] V. Dang and B. Croft. Query reformulation using anchor text. In *Proceedings of the 3rd international ACM WSDM conference*, pages 41–50. ACM, 2010.

[9] J. Gao, X. Li, D. Micol, C. Quirk, and X. Sun. A large scale ranker-based system for search query spelling correction. In *Proceedings of the 23rd international COLING conference*, pages 358–366. ACL, 2010.

[10] J. Gao, W. Yuan, X. Li, K. Deng, and J. Nie. Smoothing clickthrough data for web search ranking. In *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*, pages 355–362. ACM, 2009.

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

[12] F. Guo, C. Liu, and Y. Wang. Efficient multiple-click models in web search. In *Proceedings of the Second ACM International Conference on Web Search and Data Mining*, pages 124–131. ACM, 2009.

[13] J. Guo, G. Xu, H. Li, and X. Cheng. A unified and discriminative model for query refinement. In *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 379–386. ACM, 2008.

[14] K. Järvelin and J. Kekäläinen. Cumulated gain-based evaluation of ir techniques. *ACM Trans. Inf. Syst.*, 20(4):422–446, 2002.

[15] J.C.Borda. Mémoire sur les élections au scrution. *Histoire de l'Académie Royal des Sciences*, 1781.

[16] T. Joachims. Optimizing search engines using clickthrough data. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 133–142. ACM, 2002.

[17] M. Li, Y. Zhang, M. Zhu, and M. Zhou. Exploring distributional similarity based models for query spelling correction. In *Proceedings of the 21st international COLING conference and the 44th annual meeting of the ACL*, pages 1025–1032. ACL, 2006.

[18] F. Och. Statistical machine translation: from single-word models to alignment templates. *Germany, RW TH Aachen*, 2002.

[19] S. Robertson, H. Zaragoza, and M. Taylor. Simple bm25 extension to multiple weighted fields. In *Proceedings of the thirteenth ACM Conference on Information Knowledge Management*, page 42ÍC49, 2004.

[20] E. Sadikov, J. Madhavan, L. Wang, and A. Halevy. Clustering query refinements by user intent. In *Proceedings of the 19th international WWW conference*, pages 841–850. ACM, 2010.

[21] Y. Song and L. He. Optimal rare query suggestion with implicit user feedback. In *Proceedings of the 19th international conference on World wide web*, pages 901–910. ACM, 2010.

[22] S. Tyler and J. Teevan. Large scale query log analysis of re-finding. In *Proceedings of the 3rd international ACM WSDM conference*, pages 191–200. ACM, 2010.

[23] R. A. Wagner and M. J.Fischer. The string-to-string correction problem. *Joural of the Association for Computing Machinery*, 1974.

[24] X. Wang and C. Zhai. Mining term association patterns from search logs for effective query reformulation. In *Proceeding of the 17th ACM CIKM*, pages 479–488. ACM, 2008.

[25] J. Wen, J. Nie, and H. Zhang. Query clustering using user logs. *ACM Transactions on Information Systems*, 20(1):59–81, 2002.

[26] W. Wu, J. Xu, H. Li, and S. Oyama. Learning a robust relevance model for search using kernel methods. *Journal of Machine Learning Research*, 12:1429–1458, 2011.

[27] G. Xue, H. Zeng, Z. Chen, Y. Yu, W. Ma, W. Xi, and W. Fan. Optimizing web search using web click-through data. In *Proceedings of the thirteenth ACM international conference on Information and knowledge management*, pages 118–126. ACM, 2004.

[28] Y. Zhang, W. Chen, D. Wang, and Q. Yang. User-click modeling for understanding and predicting search-behavior. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1388–1396. ACM, 2011.

[29] Z. Zhu, W. Chen, T. Minka, C. Zhu, and Z. Chen. A novel click model and its applications to online advertising. In *Proceedings of the third ACM international conference on Web search and data mining*, pages 321–330. ACM, 2010.