Aggregating Click-Throughs of Similar Queries for Web Search

ABSTRACT

Click-through data is incredibly valuable to enhance Web search because it reflects implicit feedback from users. Most previous works leverage click-throughs of a query in ranking Web documents while they suffer from the sparsity issue of click-through data. In this paper, we propose mining the queries having similar search targets with a query and aggregating their click-throughs with the query's. We experiment with two methods to identify similar queries, i.e., one method based on co-clicks and one method based on fuzzy match of query strings, and two strategies to aggregate similar queries' click-throughs with the query's, i.e., one replacing strategy and one smoothing strategy. When we combine click-through information with a state-of-the-art ranking model based on content, experimental results indicate that leveraging click-throughs of similar queries can effectively alleviate the sparsity issue and thus improve Web search. The method based on fuzzy match significantly outperforms the baseline method that combines a query's click-throughs with the ranking model, in terms of NDCG@1, NDCG@3, and NDCG@10, and the method based on co-clicks is also significantly better than the baseline in terms of NDCG@10, if we applying the smoothing strategy.

1. INTRODUCTION

As implicit feedback from users, click-throughs are widely used by search engines in ranking Web documents. Information Retrieval community also conduct research works on this topic [11, 19, 1]. A common conclusion is that click-throughs can significantly improve retrieval effectiveness over traditional ranking models based on document content. The premise is that a query has enough click-throughs. Associated with the power-law distribution of query frequency, the number of clicks for queries is also unbalanced. A small part of popular queries have more than enough clicks while most queries have only a few clicks. This issue is known as the sparsity of click-throughs.

To address the sparsity issue, some works, such as [19, 1], extract queries for each clicked URL and allow partial match of a query in calculating relevance features based on click-throughs. As the unit of match is a word, the works cannot match "g earth" with "googleearth", or "quicktime download" with "quick time down load". Recent work [18] proposed using query similarity based on co-clicked documents to deal with the mismatch problem. The work inspires our work and we also experiment with mining similar queries based on co-clicks. In addition, we further take account of similarity based on query strings in another mining method.

In this paper, our basic idea is to mine similar queries, aggregate their click-throughs with a query's click-through offline, and then integrate the enhanced click-through information into ranking Web documents online. First, we experiment with two typical methods to mine similar queries: one is based on co-clicks and the other is additionally based on fuzzy match of query strings. Second, we aggregate click-throughs of all similar queries and use them in two strategies: one is to replace a query's click-throughs and the other is to smooth a query's click-throughs by the aggregated data. Third, we apply a simple but conventional way to fuse click information with a state-of-the-art ranking model to show the pure potentials of the compared methods.

We conduct experiments based on three-month search logs and a large scale query set and relevance assessments. Statistics show that both two mining methods increase the number of clicked URLs and the number of clicks by times. Based on evaluation results of retrieval, we have the following findings: 1) our experimental results confirm that click-throughs can dramatically improve the ranking model based content by about 10% in terms of NDCG@3 and 6% in NDCG@10. 2) Smoothing a query's click-throughs by the aggregated click-throughs of similar queries works better than replacing a query's click-throughs by the aggregated data. 3) The method based on co-clicks cannot beat the baseline method that combines a query's click-throughs with the ranking model, in terms of NDCG@1 and NDCG@3, but wins in terms of NDCG@10. 4) The method based on fuzzy match significantly outperforms the baseline method in terms of NDCG@1, NDCG@3 and NDCG@10. 5) By comparing the method based on fuzzy match and the baseline method over different query segments, we find that click-throughs of similar queries tend to benefit the queries with fewer clicks. This indicates that our goal of solving the sparsity issue is achieved.

To our best knowledge, it is the first time to compare both query similarity based on co-clicks and query similarity based on query strings to address the sparsity issue of click-throughs on such large scale data. Previous works cannot apply similarity based on query strings on large scale datasets due to complexity. We avoid the efficiency issue by comparing queries within query clusters that are obtained by the method based on co-clicks.

In the rest of the paper, we review related works and discuss the difference between our work and previous works in Section 2. In Section 3, we describe two methods to mine similar queries. Section 4 gives details on how we integrate click-throughs of similar queries into ranking functions. Then, we conduct experiments to evaluate different approaches in Section 5. Finally, we draw conclusions and discuss future work in Section 6.

2. RELATED WORK

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. [11] 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 [4, 21, 20, 8] propose several click models to infer more trustworthy relationship between URLs from search logs. For example, Zhang et al. [20] 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. [19] 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.

Nevertheless, sparsity of click-throughs is a well-known problem [7]. 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. [18] 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. [14] 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. [6] 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.

In mining similar queries, our work is related to previous works on query clustering [2, 17]. Wen et al [17] utilize the cross reference between queries and click documents to cluster queries by their intents, which expands measuring query similarity based on query strings. The scale of their experimental data is relative small. Cao et al [3] propose a clustering algorithm that is scalable to Web logs. They cluster queries into concepts to solve the sparsity issue of session logs and then deploy contextual information in user's session data to suggest better queries that fit a specific user's requirements. There are also hybrid approaches [13, 15]. Sadikov et al[13] consider a Markov model combining document and session features in predicting user intents. This work takes the advantages of both clustering and session, while random walk inference cannot be effectively finished when processing hundreds of millions of queries as a graph in our study. We adopt the clustering algorithm proposed in [3] as a method based on co-clicks. In addition, above the clustering results, we apply similarity of query strings to identify queries with more similar intent.

3. MINING SIMILAR QUERIES

In this section, we describe two methods for mining similar queries. The first method measures query similarity based on co-clicks. We adopt the implementation of query clustering proposed by previous work [3]. The second method further considers query strings in measuring similarity between queries. We propose applying the second method within clusters we obtain via the first method. Thus, we can afford the complexity of the second method and ensure quality.

3.1 A Method 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 intents.

Similar to [3], 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 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 [3] 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.

Table 1: **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)

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 [3]. 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, because 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 1 and Table 2 show four example clusters at the left side. A cluster is usually related to one concept. For example, the first cluster Table 2 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 1 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 A Method based on Fuzzy Match

To address the issue that queries in a co-clicks cluster may be less relevant to each other, we propose a method based on fuzzy match of query strings in this subsection. 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 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 misspells [5]. Furthermore, users can use difference 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 [16] in our fuzzy match method.

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 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, a large scale flooding algorithm is performed to partition the graph into connecting sub-graphs, and the nodes in each connecting sub-graph are regarded as similar queries.

Table 1 and Table 2 show four examples at the right side. Compared to the co-clicks based results, the similar queries identified by this method more likely stick to a narrow query intent. 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 sim-

Table 2: **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)

ilar 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, 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. RANKING WEB DOCUMENTS

Previous work has demonstrated that click-throughs are effective to improve Web search. Instead of complex models, we apply a simple but conventional way to fuse click-throughs with a baseline ranking function based on content. As the first work, we aim to show the pure and common potential of click-throughs of similar queries by excluding factors of ranking models. On aggregating click-throughs, we describe two strategies: a) replace 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 state-of-the-art 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 [12] from Bing Search's index. 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 [10] to combine the two lists:

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

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:

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

In additional to using click-throughs of similar queries to replace click-through of the given query, we can 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

We conduct experiments to evaluate our proposed approaches and compare them with baseline methods on large scale datasets of search logs, Web queries and assessed documents.

5.1 Experimental 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. We ignore the queries without any click because we have no enough evidence to judge whether they share similar intents with a given query. Sometimes even two queries look similar to each other, such as "SVN" and "SVM", they still have different search intents.

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.

Table 3: 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 4: 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%†

We also 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. 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) [9] as evaluation metrics of retrieval. The five relevance ratings, from 1 to 5, 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 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

We do statistics of similar queries and click-throughs over the 7,543 queries. We average the number of similar queries, the number of clicked URLs and the number of clicks. Results are shown in Table 3. 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. It is also due to the strategy of sampling queries. More frequent queries have more chance to get in. In addition, when we intersect the query set with the search logs, the common queries tend to be more popular as the original queries were selected from previous logs.

According to Table 3, the method based on 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 (CC) 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 Experimental Results

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 4.

In the table, the first two methods are traditional. We find that integrating click-through into BM25, the state-of-the-art ranking algorithm based on content, can significantly improve retrieval effectiveness in terms of all three NDCGs. And the improvements are dramatic, in particular for top performance. This confirms that click-throughs are valuable implicit feedback on relevance from users and essentially helpful for Web search, as previous works showed.

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 [3], 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 1. 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 intentions 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.

The fifth and sixth methods smooth query clicks by aggregated clicks of similar queries and then combine the smoothed clicks with BM25. As Table 4 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 2. 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 4 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. It is necessary to use the click-throughs of similar queries to smooth query clicks, rather than replacing query clicks. That is to say, the click-throughs of similar queries are complementary to the click-throughs of a query. For example, as shown in Table 5, 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.

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 1. 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 1.

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 for improving top retrieval performance. 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. For the queries with more than 135 clicks, our proposed method cannot significantly improve retrieval effectiveness of top three documents.

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, 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.

We conduct a case study on why click-throughs of similar queries can improve retrieval effectiveness for queries with different number of clicks. Some cases are shown in Table 6. 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 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 sim-

Table 5: 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)

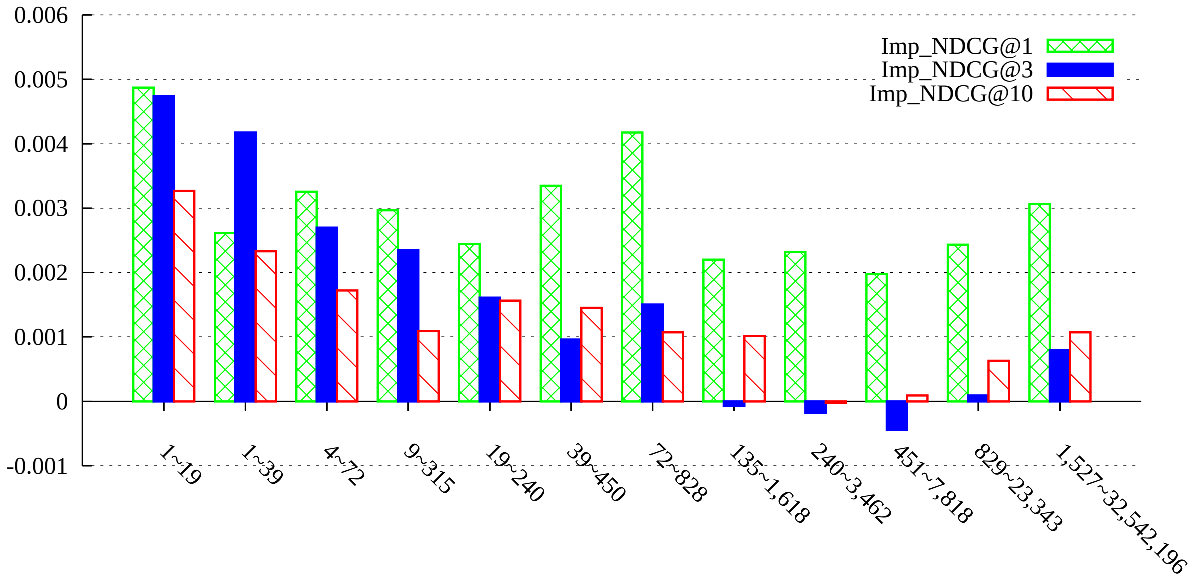

Figure 1: Analyze improvement of the (Q,FM)+BM25 method over the Q+BM25 method on different query segments divided by the total number of clicks for a query

ilar 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.

6. CONCLUSION AND FUTURE WORK

We propose to aggregate click-throughs of similar queries to address the sparsity issue. We cluster similar queries based on co-clicks, and further apply fuzzy match to identify more similar queries based on query strings within clusters. Then, we use click-throughs of similar queries to replace a query's click-throughs or smooth them. When we integrate click information into the content based ranking model BM25 by Borda's combination, we compare the two mining methods and the two strategies of using the aggregated click-throughs from similar queries on large scale datasets. 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 that combines a query's click-throughs with BM25, in terms of NDCG@1 and NDCG@3, but it works the best in terms of NDCG@10. Top retrieval effectiveness is hurt because some similar queries based on co-clicks drift from the query's topic; however, the aggregated click-throughs are still helpful in improving retrieval at deeper cutoffs. The mining method based on fuzzy match wins the baseline method in terms of NDCG@1, NDCG@3 and NDCG@10, 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.

As future works, we plan to build aggregated click-throughs into data streams and conduct online experiments. Then we are going to try more complex approaches that integrate click information into ranking functions. Our next goal is to optimize the retrieval effectiveness of leveraging aggregated click-throughs of similar queries in ranking Web documents. In addition, we will try other options in mining similar queries, for example, applying a kernel method to optimize query similarity based on co-clicks as done in [18].

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] 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.

Table 6: **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)

[3] 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.

[4] 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.

[5] 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.

[6] 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.

[7] 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.

[8] 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.

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

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

[11] 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.

[12] 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.

[13] 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.

[14] 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.

[15] 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.

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

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

[18] 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.

[19] 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.

[20] 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.

[21] 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.